

INSTITUTO TECNOLÓGICO DE CHIHUAHUA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

***“MÉTODO DE APRENDIZAJE DE MÁQUINA
EMBEBIDO BASADO EN REDES AUTOENCODER
PARA DETECCIÓN DE MICROORGANISMOS”***

TESIS

QUE PARA OBTENER EL GRADO DE

***MAESTRO EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA***

PRESENTA:

EDGAR ALEJANDRO SALAZAR GONZÁLEZ

DIRECTOR DE LA TESIS:

DR. JUAN ALBERTO RAMIREZ QUINTANA



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®



CHIHUAHUA, CHIH., OCTUBRE 2023



Chihuahua, Chih. 24 de octubre de 2023

**C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ
PRESENTE**

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada **"Método de aprendizaje de máquina embebido basado en redes autoencoder para detección de microorganismos"** dirigida por el Dr. Juan Alberto Ramírez Quintana con el siguiente contenido de capítulos:

- I Antecedentes
- II Base de datos
- III Modelo propuesto
- IV Implementación en Jetson
- V Resultados
- VI Conclusiones
- VII Referencias

ATENTAMENTE

Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

**LEONCIO OCHOA CERVANTES
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

LOC/adcs





Chihuahua, Chih. 24 de octubre de 2023

LEONCIO OCHOA CERVANTES
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno **C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ**, titulado **"Método de aprendizaje de máquina embebido basado en redes autoencoder para detección de microorganismos"** dirigido por el Dr. Juan Alberto Ramírez Quintana, ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

ATENTAMENTE

Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

Juan Ramírez

DR. JUAN ALBERTO RAMÍREZ QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL

DR. MARIO IGNACIO CHACÓN MURGUÍA
MIEMBRO DEL COMITÉ TUTORIAL

MTRA. ALMA DELIA CORRAL SÁENZ
MIEMBRO DEL COMITÉ TUTORIAL

Carlos Arzate Q.

MTRO. CARLOS ARZATE QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL





CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 25 de octubre de 2023, el que suscribe, C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ con número de control G21061144, de la Maestría en Ciencias en Ingeniería Electrónica, adscrita a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del el Dr. Juan Alberto Ramírez Quintana y cede los derechos del trabajo titulado "Método de aprendizaje de máquina embebido basado en redes autoencoder para detección de microorganismos", al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ





DECLARACIÓN DE ORIGINALIDAD

En la ciudad de Chihuahua el día 25 de octubre de 2023, el que suscribe, C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ de la Maestría en Ciencias en Ingeniería Electrónica, con número de control G21061144, adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual de la tesis titulada "Método de aprendizaje de máquina embebido basado en redes autoencoder para detección de microorganismos" bajo la dirección el Dr. Juan Alberto Ramírez Quintana; que el contenido es original y que las fuentes de información consultadas para su fundamentación están debidamente citadas y referenciadas.

C. EDGAR ALEJANDRO SALAZAR GONZÁLEZ



Chihuahua, Chihuahua, octubre de 2023

Dra. María Elena Álvarez-Buylla Roces
Directora de Conacyt

Sirva la presente para saludarla e informarle que a la fecha he obtenido el grado de **Maestro en Ciencias en Ingeniería Electrónica** en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, motivo por el cual agradezco todo el apoyo brindado por este Consejo que usted representa, con el otorgamiento de una beca que permitió dedicarme de manera exclusiva a la realización de mis estudios de posgrado y de esta manera lograr el objetivo principal del convenio establecido.

Sin otro particular quedo a sus órdenes no sin antes reiterar mi agradecimiento.
Atentamente

A handwritten signature in cursive script, reading "Edgar Salazar".

Edgar Alejandro Salazar González

AGRADECIMIENTOS

En este documento de tesis quiero expresar mi más profundo agradecimiento a todas las personas e instituciones que me apoyaron durante la realización de la Maestría en Ciencias en Ingeniería Electrónica.

En primer lugar, agradezco a mis padres y a mi hermano por su amor incondicional, su confianza y su aliento en cada momento de este camino. Ellos son mi inspiración y mi motivación para seguir adelante.

En segundo lugar, agradezco a mis compañeros de Laboratorio y de otras áreas de Posgrado por su amistad, su colaboración y su ayuda en los aspectos técnicos y personales de este proyecto. Ellos hicieron que el trabajo fuera más divertido y enriquecedor.

En tercer lugar, agradezco a mi director de tesis, el Dr. Juan Alberto Ramírez Quintana por su guía, su paciencia y su exigencia académica. Él me enseñó a pensar críticamente y a resolver problemas complejos con creatividad e inteligencia artificial. Su experiencia y conocimiento fueron fundamentales para el éxito de esta investigación.

Por último, agradezco al Instituto Tecnológico de Chihuahua por brindarme la oportunidad de estudiar esta Maestría, por proveerme de los recursos necesarios y por fomentar un ambiente de excelencia y calidad educativa.

A todos ellos, mi eterno reconocimiento y gratitud.

RESUMEN

MÉTODO DE APRENDIZAJE DE MÁQUINA EMBEBIDO BASADO EN REDES AUTOENCODER PARA DETECCIÓN DE MICROORGANISMOS

Edgar Alejandro Salazar González

Maestría en Ciencias en Ingeniería Electrónica

División de Estudios de Posgrado e Investigación del

Instituto Tecnológico de Chihuahua

Chihuahua, Chih. 2023

Director de Tesis: Dr. Juan Alberto Ramírez Quintana

La microbiología es una ciencia que desempeña un papel vital en el diagnóstico de enfermedades mediante el análisis de bacterias, levaduras y protozoos. Una técnica esencial para la microbiología es el conteo de microorganismos en cultivos ya que permite detectar y evaluar la presencia de agentes infecciosos. Esto es fundamental para el seguimiento de enfermedades, monitoreo de la carga microbiana en muestras biológicas, evaluar la respuesta al tratamiento, garantizar una recuperación efectiva e investigación ya que probar nuevos agentes antimicrobianos y técnicas de desinfección requiere el conteo de microorganismos. Una herramienta que se ha utilizado exitosamente en el campo de la microbiología es la inteligencia artificial, en particular las redes neuronales tipo *autoencoders*, ya que permiten la detección y conteo conciso de microorganismos en muestras microscópicas. Basándose en estos argumentos, en esta tesis se propone una arquitectura de red totalmente convolucional para facilitar el análisis de imágenes y por medio de una estructura tipo *autoencoder* para la codificación y decodificación, resaltando bacterias y levaduras en cultivos microscópicos utilizados para cuantificar grupos de microorganismos patógenos tomados por medio de imágenes digitales.

Los experimentos se hicieron en la base de datos llamada "Imágenes Reales de Bacterias y Levaduras" (BYRI en inglés) presentada en esta tesis, se compone de 3 clases: *Escherichia Coli*,

Staphylococcus Aureus y Candida Albicans. La base de datos tiene cerca de 500 imágenes por clase generadas por la Universidad Autónoma de Chihuahua (UACH).

La red propuesta se denominó Red de Conteo de Microorganismos (M-Net), que tiene 575 parámetros y una exactitud del 95% en el conteo de microorganismos. M-Net se implementó en un sistema GPU embebido NVIDIA Jetson TX2 donde los recursos son limitados en comparación de una computadora de escritorio, la red propuesta como se mencionó anteriormente cuenta con un número reducido de parámetros, esto se traduce en un bajo coste computacional, este modelo podría implementarse en un sistema de visión en microscopio facilitando la identificación y seguimiento de microorganismos, estructuras subcelulares y otros elementos importantes y todo por medio de *hardware* discreto.

CONTENIDO

“MÉTODO DE APRENDIZAJE DE MÁQUINA EMBEBIDO BASADO EN REDES AUTOENCODER PARA DETECCIÓN DE MICROORGANISMOS”	1
AGRADECIMIENTOS.....	v
RESUMEN.....	vi
CONTENIDO.....	viii
LISTA DE FIGURAS	xi
LISTA DE TABLAS.....	xiii
INTRODUCCIÓN	xiv
1 ANTECEDENTES	1
1.1 Caldo nutritivo.....	1
1.1.1 Bacteria.....	2
1.1.2 Levadura.....	2
1.2 Microorganismos de interés	2
1.2.1 Escherichia coli	3
1.2.2 Staphylococcus aureus.....	3
1.2.3 Candida albicans.....	4
1.2.4 Microbiología Analítica.....	4
1.3 Aprendizaje de máquina	5
1.3.1 Aprendizaje supervisado	5
1.3.2 Aprendizaje no supervisado	6
1.3.3 Aprendizaje por refuerzo	6
1.4 Aprendizaje profundo.....	7
1.4.1 Redes neuronales artificiales	7
1.5 Modelo totalmente convolucional	8
1.5.1 Capa convolucional	9
1.5.2 ReLU	10
1.5.3 Leaky- ReLU	11
1.5.4 Capa de reducción de muestreo (<i>Pooling</i>).....	11
1.5.5 Max-pooling	12
1.5.6 Upsampling	12
1.5.7 Descripción de funcionamiento del <i>autoencoder</i>	12
1.6 Redes seleccionadas para el análisis.....	14
1.6.1 ResNet	14

1.6.2	U-Net.....	15
1.6.3	MobileNet.....	16
2	BASE DE DATOS	17
2.1	Bases de datos en análisis clínico	17
2.2	Descripción de la base de datos	19
2.3	Aumento de datos	20
2.3.1	Inversión de la imagen.....	22
2.3.2	Inversión en el eje de la abscisa “x”	22
2.3.3	Inversión en el eje de la abscisa “y”	23
2.3.4	Inversión en el eje de la abscisa “xy”	23
3	MODELO PROPUESTO	24
3.1	M-Net V1	24
3.1.1	Muestras de resultados del modelo.....	27
3.2	M-Net V2	28
3.2.1	Muestras de resultados del modelo.....	30
3.3	M-Net V3	31
3.3.1	<i>Depth-wise convolution</i>	32
3.3.2	<i>Depth-wise Separable Convolution</i>	33
3.3.3	Estructura del modelo M-Net V3	35
3.3.4	Muestras de resultados del modelo.....	38
3.4	Función de pérdida	39
4	IMPLEMENTACIÓN EN JETSON	41
4.1	Conceptos de sistemas embebidos.....	41
4.1.1	GPU embebida	41
4.1.2	CISC.....	41
4.1.3	RISC.....	42
4.2	Jetson TX2.....	44
4.3	Implementación de M-Net V3 en la Jetson TX2.....	45
4.3.1	Funcionamiento en tarjeta de desarrollo Nvidia Jetson TX2.....	45
4.3.2	CPU	46
4.3.3	GPU	47
4.3.4	Diferencia entre usar <i>Tensorflow</i> y <i>Onnx</i>	50
5	RESULTADOS	52
5.1	Proceso de generación de conteo.....	52
5.1.1	Procesamiento de los datos.....	52
5.1.2	Umbralización	54
5.1.3	Análisis cualitativo	56

5.2 Comparación de N-Net con otras FCN.....	61
5.3 Prueba de hipótesis	64
5.3.1 Prueba de Lilliefors	64
5.3.2 Prueba de <i>Wilcoxon</i>	74
6 CONCLUSIONES.....	78
6.1 Trabajo realizado	78
6.2 Trabajo a futuro	79
7 REFERENCIAS	80

LISTA DE FIGURAS

Figura 1.1 a) Escherichia coli b) Staphylococcus aureus c) Candida albicans.	3
Figura 1.2 Arquitectura sencilla de una red neuronal.	8
Figura 1.3 Diagrama de un autoencoder.	9
Figura 1.4 Ejemplo de convolución en imagen RGB [20].	10
Figura 1.5 Modelo ResNet.	15
Figura 1.6 Modelo U-Net.	15
Figura 1.7 Modelo MobileNet.	16
Figura 2.1 Ejemplos de ground truth generados de cada clase.	18
Figura 2.2 Muestra de imágenes de base de datos BYRI a) Candida Albicans b) Escherichia Coli c) Staphylococcus Aureus.	20
Figura 2.3 Muestras de GT generadas (no están relacionadas a la Figura 2.2). a) Candida Albicans b) Escherichia Coli c) Staphylococcus Aureus.	21
Figura 2.4 Ejemplo de aumento de datos a) imagen original, b) rotación de 180° eje 'x', c) inversión en eje 'y', d) rotación de 180° e inversión en eje 'y' y en eje 'x'.	22
Figura 3.1 Representación sintetizada de M-Net V1 y V2.	25
Figura 3.2 Convoluciones de M-Net V1.	25
Figura 3.3 Arquitectura del modelo de autoencoder diseñado (M-Net V1).	25
Figura 3.4 Resultados del modelo V1.	28
Figura 3.5 Convoluciones utilizadas en M-Net V2.	29
Figura 3.6 Arquitectura del modelo de autoencoder diseñado (M-Net V2).	29
Figura 3.7 Resultados del modelo V2.	31
Figura 3.8 Ejemplo de cómo se realiza una convolución profunda en una imagen RGB [20].	33
Figura 3.9 Ejemplo de una convolución separable en profundidad en una imagen RGB [20].	35
Figura 3.10 Representación sintetizada de M-Net V3.	36
Figura 3.11 Convoluciones utilizadas y orden de acomodo en M-Net V3.	36
Figura 3.12 Arquitectura del modelo de autoencoder diseñado (M-Net V3).	36
Figura 3.13 Resultados del modelo V3.	39
Figura 4.1 Resultados con M-Net V3 a) Salida del modelo b) Salida del modelo binarizada c) Comparación de la entrada con los resultados encontrados (encerrados en rojo).	46
Figura 4.2 Tiempo de generación de resultados usando CPU.	47
Figura 4.3 Tabla de consumo energético de tarjeta de desarrollo Nvidia Jetson TX2 mientras corre el modelo M-Net V3 obtenido con jtop (CPU).	47
Figura 4.4 Tiempo de generación de resultados usando GPU.	48
Figura 4.5 Tabla de consumo energético de tarjeta de desarrollo Nvidia Jetson TX2 mientras corre el modelo M-Net V3 obtenido con jtop (GPU).	49
Figura 4.6 Comparación de resultados con 2 formatos distintos (Tensorflow y onnx), los casos registrados como verdaderos son resultados iguales entre formatos, los resultados evaluados como falsos son resultados distintos.	50
Figura 5.1 Resultados obtenidos en el procedimiento de Predicción (Clase Candida Albicans), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), Resultado de la imagen de ejemplo (M-Net V3).	52
Figura 5.2 Resultados obtenidos en el procedimiento de Predicción (Clase Escherichia coli), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3).	53
Figura 5.3 Resultados obtenidos en el procedimiento de Predicción (Clase Staphylococcus aureus), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3).	53
Figura 5.4 Resultados obtenidos en el procedimiento de Umbralización (Clase Candida Albicans), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3).	54

Figura 5.5 Resultados obtenidos en el procedimiento de Umbralización (Clase Escherichia coli), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3).....	55
Figura 5.6 Resultados obtenidos en el procedimiento de Umbralización (Clase Staphylococcus aureus), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3).....	55
Figura 5.7 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase candida albicans).	57
Figura 5.8 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase escherichia coli).	58
Figura 5.9 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase staphylococcus aureus).....	59
Figura 5.10 Comportamiento de hoja de normalidad, clase Candida albicans (M-Net V1 (Otsu)).....	65
Figura 5.11 Comportamiento de hoja de normalidad, clase Candida albicans (M-Net V2 (Otsu)).....	66
Figura 5.12 Comportamiento de hoja de normalidad, clase Candida albicans (M-Net V3 (Umbral))	66
Figura 5.13 Comportamiento de hoja de normalidad, clase Candida albicans (MobileNet (Umbral))	67
Figura 5.14 Comportamiento de hoja de normalidad, clase Candida albicans (ResNet (Umbral))	67
Figura 5.15 Comportamiento de hoja de normalidad, clase Candida albicans (U-Net (Umbral)).....	68
Figura 5.16 Comportamiento de hoja de normalidad, clase Escherichia coli (M-Net V1 (Otsu)).....	68
Figura 5.17 Comportamiento de hoja de normalidad, clase Escherichia coli (M-Net V2 (Otsu)).....	69
Figura 5.18 Comportamiento de hoja de normalidad, clase Escherichia coli (M-Net V3 (Umbral))	69
Figura 5.19 Comportamiento de hoja de normalidad, clase Escherichia coli (MobileNet (Umbral))	70
Figura 5.20 Comportamiento de hoja de normalidad, clase Escherichia coli (ResNet (Umbral)).....	70
Figura 5.21 Comportamiento de hoja de normalidad, clase Escherichia coli (U-Net (Umbral))	71
Figura 5.22 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (M-Net V1 (Otsu)).....	71
Figura 5.23 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (M-Net V2 (Otsu)).....	72
Figura 5.24 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (M-Net V3 (Umbral))	72
Figura 5.25 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (MobileNet (Umbral))	73
Figura 5.26 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (ResNet (Umbral))	73
Figura 5.27 Comportamiento de hoja de normalidad, clase Staphylococcus aureus (U-Net (Umbral)).....	74
Figura 5.28 Resultado ideal que se busca obtener de los modelos utilizados con prueba Wilcoxon.	75
Figura 5.29 Comportamiento con clase Candida albicans (para las 3 versiones de M-Net)	75
Figura 5.30 Comportamiento con clase Candida albicans (Modelos del estado del arte)	76
Figura 5.31 Comportamiento con clase Escherichia coli (para las 3 versiones de M-Net)	76
Figura 5.32 Comportamiento con clase Escherichia coli (Modelos del estado del arte)	76
Figura 5.33 Comportamiento con clase Staphylococcus aureus (para las 3 versiones de M-Net)	77
Figura 5.34 Comportamiento con clase Staphylococcus aureus (Modelos del estado del arte)	77

LISTA DE TABLAS

Tabla 4.1 Resumen de arquitecturas CISC y RISC.	43
Tabla 4.2 Especificaciones de PC utilizada para el entrenamiento.	44
Tabla 4.3 Comparación de resultados entre CPU y GPU [44] [45] [46]	49
Tabla 4.4 Resultados en donde el modelo de Tensorflow y Onnx generan resultados distintos (algunos ejemplos).	51
Tabla 5.1 Resultados obtenidos de conteo empleando factor humano y modelos distintos (Candida albicans). ..	60
Tabla 5.2 Resultados obtenidos de conteo empleando factor humano y modelos distintos (Escherichia coli).	60
Tabla 5.3 Resultados obtenidos de conteo empleando factor humano y modelos distintos (Staphylococcus aureus).	61
Tabla 5.4 Comparación entre modelos M-Net, resultados finales del entrenamiento.	63
Tabla 5.5 Comparación entre modelos del estado del arte, resultados finales del entrenamiento.	63

INTRODUCCIÓN

El análisis clínico y las pruebas de laboratorio son algunos de los temas más importantes en consultas de atención médica ya que permiten conocer los indicadores relacionados a la compatibilidad del cuerpo humano con la vida y salud. A través de una prueba de laboratorio es posible diagnosticar el estado de salud de un paciente al interpretar los niveles de hierro, colesterol, bilirrubina, glucosa y muchos otros tipos de elementos presentes en dicha muestra. Estas pruebas, dada su importancia, seguras, eficaces, asequibles y sin discriminación alguna. Sin embargo, no todas las personas tienen acceso a estos servicios debido a varios desafíos como: barreras económicas, desigualdades geográficas, falta de conciencia y educación, barreras culturales, barreras lingüísticas y falta de infraestructura, no todas las personas tienen acceso a ser diagnosticados con base en estas pruebas.

Una solución prometedora para superar algunas de estas barreras y mejorar el acceso a los servicios de análisis clínicos y pruebas de laboratorio es la aplicación de redes neuronales artificiales en telemedicina [1] [2]. Por ejemplo, mediante el uso de dispositivos embebidos y sistemas basados en la nube, los datos recopilados de los pacientes pueden ser procesados de forma remota utilizando redes neuronales y enviados al personal médico. Esto amplía el acceso a los servicios de análisis clínicos en áreas rurales o con recursos limitados, donde la infraestructura de laboratorios convencionales puede ser escasa.

Existen distintas aplicaciones en el área médica y de investigación cuando se utilizan redes neuronales artificiales con sistemas embebidos y telemedicina. Por ejemplo, en [3] se hace uso de la Inteligencia Artificial (IA) con el fin de identificar plantas medicinales para agruparlas de acuerdo a las necesidades del usuario. Otra utilidad se aprecia en [4], donde se clasifica videos fetoscópicos, los cuales son procedimientos para acceder al feto durante el embarazo con herramientas para diferenciar y clasificar anastomosis. Las redes neuronales también se aplican en la detección de cáncer de mama en imágenes histológicas, por ejemplo, en [5] donde se implementa por medio de *deep learning* un clasificador usando CNN para detectar cáncer de mama en imágenes termográficas. Como estos trabajos, se pueden mencionar múltiples ejemplos del uso de redes neuronales con sistemas embebidos y telemedicina aplicados a imágenes médicas. Por ejemplo, una aplicación interesante es el conteo de microorganismos en

una muestra que se utiliza para conocer el estado de salud de una persona, ya que la cantidad de microorganismos en una muestra de cultivo indica diversos aspectos de la salud de un paciente. Por ello, en el proyecto *Algoritmos de visión artificial para realizar análisis de imágenes digitales para diagnóstico clínico parasitológico, urológico, hematológico y en muestras histológicas de patología* que se realiza entre el Tecnológico Nacional de México – Instituto Tecnológico de Chihuahua (ITCH) y la Facultad de Medicina y Ciencias Biomédicas (FMCB) de la Universidad Autónoma de Chihuahua, se busca cuantificar microorganismos en diferentes muestras microscópicas para conocer si un sujeto padece de una enfermedad infecciosa. Para contribuir con los esfuerzos de este proyecto, en esta tesis se propone una arquitectura de red totalmente convolucional para la detección y conteo de bacterias y levaduras en cultivos microscópicos. La arquitectura se denominó Red de Conteo de Microorganismos (M-Net) y se desarrollaron tres versiones: M-Net V1, M-Net V2 y M-Net V3. Los experimentos se realizaron en una base de datos que se denominó Imágenes Reales de Bacterias y Levaduras (BYRI por sus siglas en inglés) y se compone de 1502 imágenes divididas en tres clases. Entre estas 1502 imágenes, 500 son de cultivos de *Escherichia Coli*, 506 de cultivos de *Staphylococcus Aureus* y 496 de cultivos de *Candida Albicans*.

La presente tesis está dividida de la siguiente manera: el **capítulo I** presenta los antecedentes encontrados en la literatura. El **capítulo II** describe la base de datos utilizada y las técnicas usadas para utilizarse en el entrenamiento del modelo. El **capítulo III** explica a profundidad del modelo propuesto para resolver el problema y el **capítulo IV** se muestra el modelo en el hardware propuesto y sus resultados. El **capítulo V** se presenta los resultados del modelo y, por último, el **capítulo VI** describe las conclusiones.

1 ANTECEDENTES

El uso de la inteligencia artificial ha revolucionado el procesamiento de imágenes médicas en los últimos años. Esto se debe a que el aprendizaje profundo ha demostrado un gran potencial para el análisis y la interpretación automatizada de imágenes médicas. Dos enfoques populares dentro del aprendizaje profundo son los *autoencoders* y las redes totalmente convolucionales. Los *autoencoders* son redes neuronales artificiales que se utilizan para aprender una representación compacta y significativa de los datos de entrada. Tienen una etapa de codificación que comprime la imagen de entrada, y una etapa de decodificación que reconstruye la entrada a partir de la representación comprimida. Los *autoencoders* son eficaces en el procesamiento de imágenes médicas, especialmente en tareas de eliminación de artefactos, restauración y mejora en la calidad de las imágenes. Por otro lado, las redes totalmente convolucionales (*Fully Convolutional Networks*, FCN) son *autoencoders* [6] con neuronas convolucionales diseñados para tareas de segmentación. Esto les permite procesar imágenes y generar mapas de activación que indican la presencia y ubicación regiones que representen objetos de interés.

Las FCN han sido ampliamente utilizadas en el procesamiento de imágenes médicas para la segmentación precisa de estructuras anatómicas y la detección de patologías. Por ello, en este capítulo se presentan conceptos básicos que ayudarán a mejorar la comprensión de la investigación. Para conocer los conceptos relacionados con los microorganismos, las secciones **1.1** y **1.2** dan una descripción del caldo nutritivo y los microorganismos de interés en la investigación. Para abordar los temas de FCN, las secciones **1.3** a **1.5** dan una breve descripción de las redes neuronales totalmente convolucionales. Finalmente, la sección **1.6** da una descripción de las redes preseleccionadas para el análisis.

1.1 Caldo nutritivo

El caldo nutritivo (conocido como medio de cultivo), es una solución líquida que proporciona los nutrientes necesarios para el crecimiento y la reproducción de microorganismos en un entorno de laboratorio. Se utilizan en microbiología y biología molecular para analizar células, bacterias, levaduras u otros microorganismos. Los caldos de cultivo pueden tener ingredientes como extractos de carne, extractos de levadura, peptona (proteínas hidrolizadas),

glucosa, sales inorgánicas y otros componentes que varían según el tipo de microorganismo que se desee cultivar [7] [8].

1.1.1 Bacteria

Las bacterias son microorganismos procariotas que presentan un tamaño de pocos micrómetros (por lo general entre 0,5 y 5 μm de longitud) [8]. Existen diferentes tipos de bacterias que varían en forma, tamaño y características metabólicas. Algunas tienen forma de esfera (cocos), otras tienen forma de bastón (bacilos) y otras tienen forma de espiral (espirilos). Además, las bacterias pueden tener estructuras adicionales, como flagelos para la locomoción y pili para la adhesión a superficies [8].

Las bacterias desempeñan un papel crucial en la naturaleza y en la vida humana. Algunas son benéficas y desempeñan funciones importantes, como la descomposición de la materia orgánica, la producción de alimentos fermentados, la síntesis de vitaminas y la participación en procesos biotecnológicos. Sin embargo, también existen bacterias patógenas que pueden causar enfermedades infecciosas en los seres humanos y otros organismos [8].

1.1.2 Levadura

Se denomina levadura a cualquiera de los hongos microscópicos que son predominantemente unicelulares en su ciclo de vida. Generalmente, se reproducen asexualmente por gemación o bipartición o con estados sexuales que no están adjuntos a un esporocarpo (cuerpo fructífero). Algunas levaduras como *Candida* pueden formar cadenas de células conectadas en ciernes, conocidas como pseudohifas. Las levaduras, especialmente *Candida albicans* causan infecciones en los seres humanos llamadas candidiasis y pueden afectar diferentes áreas del cuerpo como piel, uñas, boca, genitales y órganos internos [9].

1.2 Microorganismos de interés

Para este proyecto de tesis, se van a analizar los siguientes microorganismos:

- *Candida albicans*.
- *Escherichia coli*.
- *Staphylococcus aureus*.

Dichos microorganismos fueron seleccionados para estudio debido a que pueden ocasionar problemas a la salud en ciertas condiciones. A continuación, se da una descripción de cada microorganismo, así como el impacto que puede tener en la salud de un ser humano.

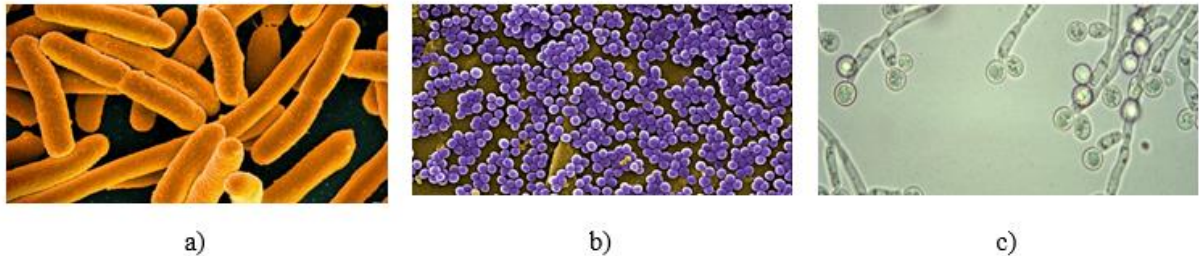


Figura 1.1 a) *Escherichia coli* b) *Staphylococcus aureus* c) *Candida albicans*.

1.2.1 *Escherichia coli*

Escherichia coli (EC) (Figura 1.1a) es un tipo de bacteria que vive en el intestino. La mayoría de las variantes de *Escherichia coli* no causan problemas y forman parte de la flora intestinal. Sin embargo, algunas variantes de esta bacteria pueden producir enfermedades y causar diarrea. La bacteria EC causa una diarrea hemorrágica, insuficiencia renal y en casos extremos hasta la muerte. Esto, en general, ocurre en niños y en adultos con sistemas inmunitarios debilitados [10]. La bacteria se transmite al ser humano por el consumo de alimentos contaminados, como productos de carne poco cocida, leche cruda, hortalizas y semillas germinadas crudas contaminadas [10].

1.2.2 *Staphylococcus aureus*

Staphylococcus aureus (SA) (Figura 1.1b) es una bacteria conocida por ser la principal causa de infecciones sanguíneas en entornos hospitalarios en todo el mundo. Presenta un aumento en la resistencia a diversos factores patógenos y virulentos. Además, produce una variedad de proteínas que se encuentran en la superficie bacteriana y se conocen como moléculas de la matriz adhesiva (MSCRAMM). Estas proteínas desempeñan un papel crucial en la colonización e invasión de células del huésped, así como en la formación de biopelículas. Estos mecanismos de patogenicidad y virulencia permiten que la bacteria persista tanto en el huésped como en el entorno, sobreviviendo a factores adversos, al sistema inmunológico y a los tratamientos antimicrobianos [11].

Los síntomas típicos de la intoxicación estafilocócica incluyen náuseas, vómitos, dolor de estómago y abdominal que se manifiestan rápidamente (en 1-6 horas) después de ingerir alimentos contaminados, y pueden durar de uno a dos días. Esta infección es especialmente preocupante en personas con sistemas inmunológicos débiles, como bebés, niños menores de 5 años, personas mayores de 60 años, pacientes con cáncer, diabetes, VIH, entre otros, debido a la deshidratación vinculada a los síntomas gastrointestinales [11].

1.2.3 *Candida albicans*

Candida albicans (CA) (Figura 1.1c) es un hongo dimórfico perteneciente al Phylum Ascomycota, que presenta pseudohifas, hifas y blastoconidios subesféricos (3-8 x 2-7 μm). Esta levadura está asociada a animales de sangre caliente y coloniza boca, los tractos digestivos, genitales, vías respiratorias humanas, piel, uñas y membranas mucosas. La infección por CA es común en pacientes inmunodeprimidos [9]. Las infecciones por *Candida* afectan diferentes partes del cuerpo de diversas formas, como manchas blancas en la boca, dolor al tragar, vaginitis en mujeres, picazón y erupciones en la piel, y, en casos graves, el riesgo de poner en peligro la vida [9].

1.2.4 Microbiología Analítica

En microbiología, dos técnicas esenciales son el "Ensayo de Inhibición Microbiológica" y el "Conteo de Microorganismos en Cámara de Neubauer". La primera evalúa la capacidad de sustancias para detener el crecimiento de microorganismos, mientras que la segunda mide la concentración microbiana en muestras líquidas. Estas técnicas son vitales en investigación y la industria farmacéutica, permitiendo comprender y controlar la dinámica microbiana en diversos contextos. En este análisis, ahondaremos en su importancia y aplicaciones en microbiología.

Ensayo de Inhibición Microbiológica: El ensayo de inhibición microbiológica es una técnica utilizada en microbiología para evaluar la capacidad de una sustancia (como un fármaco o un agente químico) para inhibir o detener el crecimiento de microorganismos, como bacterias, hongos u otros patógenos. Este ensayo se realiza exponiendo los microorganismos a diferentes concentraciones de la sustancia en cuestión y observando su efecto sobre el crecimiento o la

supervivencia de los microorganismos. Se utiliza comúnmente en la investigación y en la industria farmacéutica para determinar la eficacia de agentes antimicrobianos, como antibióticos [12].

Conteo de Microorganismos en Cámara de Neubauer: El conteo de microorganismos en cámara de Neubauer es una técnica de laboratorio utilizada para determinar la concentración o densidad de microorganismos en una muestra líquida, como una suspensión de bacterias o células en un cultivo. La cámara de Neubauer es una cuadrícula de microscopio especialmente diseñada que permite contar microorganismos bajo un microscopio. La muestra se coloca en la cámara de Neubauer, y los microorganismos se cuentan observando bajo el microscopio cuántos microorganismos se encuentran en una serie de cuadrados conocidos. Este método es útil en microbiología y biología celular para cuantificar la densidad de microorganismos en cultivos celulares o para determinar la concentración de células en una muestra biológica [12].

1.3 Aprendizaje de máquina

El aprendizaje de máquina (*Machine Learning*) es un campo de las ciencias de la computación que desarrolla métodos para que un sistema aprenda a resolver una tarea específica a partir de los datos. A diferencia de la programación explícita, el aprendizaje de máquina permite a la computadora aprender con base en la información presentada para que los modelos resultantes sean capaces de generalizar el comportamiento de los datos que se le presentan [13]. Para dicha generalización, existen diferentes tipos de aprendizajes, los más generales son el supervisado, no supervisado y por refuerzo. A continuación, se describirán cada una de estas formas de aprendizaje.

1.3.1 Aprendizaje supervisado

El aprendizaje supervisado es un enfoque en el campo del aprendizaje automático donde se entrena un modelo utilizando datos etiquetados. En este proceso, se proporciona al modelo un conjunto de ejemplos de entrenamiento junto con las etiquetas correspondientes, que son las salidas deseadas o correctas. El objetivo del aprendizaje supervisado es que el modelo aprenda a mapear correctamente las entradas a las salidas esperadas [14]. Durante el entrenamiento, el modelo examina los datos en búsqueda de patrones y relaciones entre las características de

entrada y las etiquetas. Utilizando algoritmos y técnicas específicas, el modelo ajusta sus parámetros internos para minimizar la discrepancia entre las salidas predichas y las salidas reales [15]. Una vez entrenado, el modelo puede recibir nuevas entradas y hacer predicciones basadas en su conocimiento adquirido durante el entrenamiento. El rendimiento del modelo se evalúa comparando las predicciones correctas de un conjunto de datos de prueba [14].

1.3.2 Aprendizaje no supervisado

El aprendizaje no supervisado es una rama del aprendizaje de máquina en que el modelo se entrena sin la guía explícita de etiquetas o respuestas previas. A diferencia del aprendizaje supervisado, el modelo debe encontrar patrones o estructuras intrínsecas en los datos por sí mismo. El objetivo principal del aprendizaje no supervisado es descubrir la estructura oculta o representaciones latentes de los datos, sin tener información previa sobre las clases o categorías a las que pertenecen dichos datos. Es una tarea desafiante pero valiosa, ya que permite identificar relaciones, agrupaciones o características comunes que podrían no ser evidentes a simple vista.

1.3.3 Aprendizaje por refuerzo

El aprendizaje por refuerzo es una rama del aprendizaje de máquina que se centra en el desarrollo de agentes capaces de aprender a tomar decisiones a través de la interacción con un entorno. En este paradigma, el modelo no recibe datos de entrenamiento con etiquetas, ni tiene una función de costo definida. En cambio, el agente aprende a través de la retroalimentación que recibe del entorno en forma de recompensas o penalizaciones. El proceso de entrenamiento por refuerzo se asemeja a cómo un ser humano puede aprender a realizar una tarea a través de la experiencia y la práctica. El modelo realiza una serie de acciones en un entorno y recibe una señal de recompensa o castigo en función de la calidad de sus acciones. El objetivo del agente es maximizar la acumulación de recompensas a lo largo del tiempo, lo que a menudo se conoce como "recompensa acumulada" o "retorno".

1.4 Aprendizaje profundo

El aprendizaje profundo (*Deep Learning*) es un conjunto de modelos y algoritmos de *Machine Learning* que modelan abstracciones de alto nivel con redes cuya arquitectura consta de múltiples capas con transformaciones lineales y no lineales. La Figura 1.2 muestra un ejemplo de una red de *Deep Learning* conformado por la señal de entrada donde se reciben los datos a procesar, la capa oculta la cual realiza las tareas de extracción de características y se tiene la salida donde se realiza la predicción del modelo.

Una red neuronal artificial (RNAs) es un modelo de procesamiento de información inspirado en el funcionamiento del cerebro humano. Consiste en una red de nodos interconectados que procesan datos y aprenden patrones para realizar tareas como reconocimiento de patrones, toma de decisiones o resolución de problemas. Las RNAs se utilizan en aprendizaje de máquina y son esenciales en aplicaciones como el reconocimiento de voz, la visión por computadora y el procesamiento de lenguaje natural [14].

Las redes de *Deep Learning* más exitosas son las Redes Neuronales Convolucionales, que han sido aplicadas a campos como visión por computadora, reconocimiento de voz, reconocimiento de señales de audio, y análisis de imágenes. Además, han demostrado resultados importantes en gran cantidad de tareas [14].

1.4.1 Redes neuronales artificiales

Las redes neuronales artificiales son modelos inspirados en la sofisticada funcionalidad de los cerebros humanos donde cientos de miles de millones de neuronas interconectadas procesan información [16]. Las investigaciones en torno al estudio de las redes neuronales artificiales buscan modelar la forma del procesamiento de la información en sistemas nerviosos biológicos, que son sistemas altamente complejos, no lineales y paralelizados [14].

En términos sencillos, las RNAs pueden realizar varias operaciones simultáneamente, mientras que las computadoras que trabajan de manera secuencial (solo realizan una operación a la vez). En este sentido las RNAs son procesadores de información de distribución paralela, construida por unidades de procesamiento sencillas, llamadas neuronas [14]. De esta manera, una RNA es un modelo computacional que utiliza neuronas artificiales interconectadas para aprender y resolver problemas. Para que la RNA logre aprender cuando es un modelo profundo,

debe de llevar un entrenamiento supervisado o no supervisado que puede requerir hasta millones de datos y para ello existen varios modelos de aprendizaje. Las RNA constan de una capa de entrada de neuronas, una o varias capas ocultas de neuronas, y una capa de salida de neuronas como se muestra en la Figura 1.2.

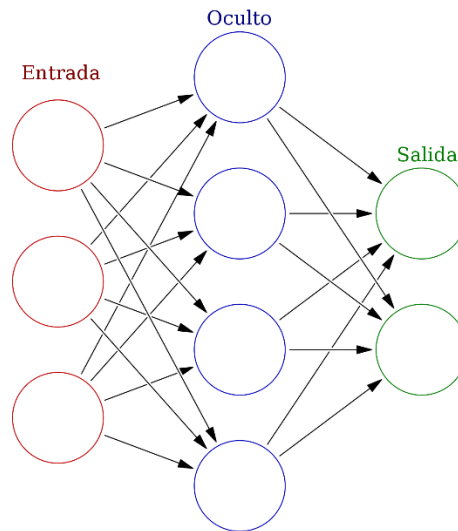


Figura 1.2 Arquitectura sencilla de una red neuronal.

1.5 Modelo totalmente convolucional

Redes neuronales completamente conectadas (FCN - Fully Connected Networks): Las redes neuronales completamente conectadas son un tipo de arquitectura de redes neuronales artificiales donde cada neurona de una capa está conectada a todas las neuronas de la capa siguiente. Esto significa que cada entrada se conecta directamente con cada neurona de la capa oculta. Las redes FCN son adecuadas para tareas de procesamiento de datos tabulares o estructurados, pero pueden tener problemas al procesar datos como imágenes debido a la pérdida de información espacial [17].

Redes neuronales convolucionales (CNN - Convolutional Neural Networks): Las redes neuronales convolucionales son un tipo especializado de red neuronal diseñado para procesar datos que tienen una estructura espacial, como imágenes o señales de audio. En lugar de conectar cada neurona de entrada con todas las neuronas de la siguiente capa, las CNN utilizan capas de convolución que aplican filtros para detectar características locales, como bordes o

texturas, en la entrada. Esto permite que las CNN capturen patrones espaciales y jerarquías de características en datos como imágenes de una manera más eficiente que las redes FCN. Las CNN son ampliamente utilizadas en tareas de visión por computadora, reconocimiento de imágenes y procesamiento de video [18] [19].

En la Figura 1.3 se muestra un ejemplo de la arquitectura de una FCN con cada una de sus capas.

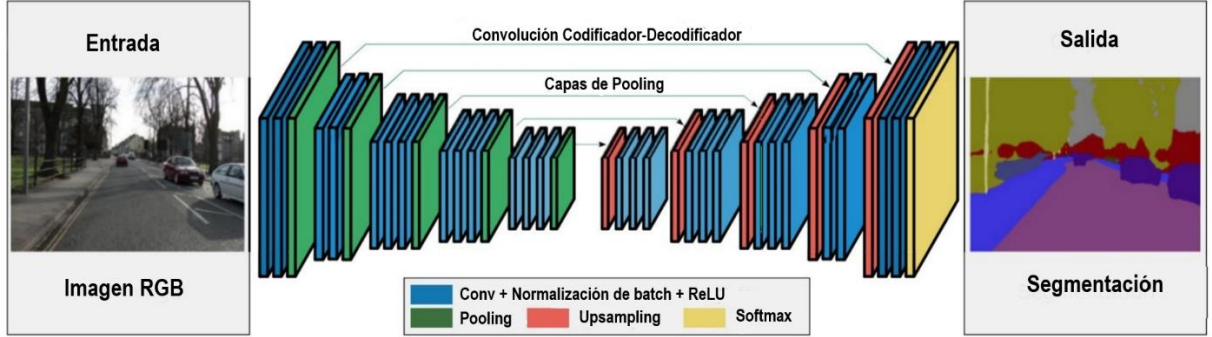


Figura 1.3 Diagrama de un autoencoder.

1.5.1 Capa convolucional

Esta capa extrae características abstractas mediante *kerneles* de convolución, dados por:

$$S_l(i, j) = (Y_{l-1} * K)(i, j) = \sum_m \sum_n Y_{l-1}(m, n) \cdot K(i - m, j - n) \quad (1.1)$$

Donde:

- $S_l(i, j)$ es el valor resultante de la convolución en la posición (i, j) del mapa de características de salida.
- $Y_{l-1}(m, n)$ es la salida de la capa anterior.
- (m, n) es el valor de píxel en la posición (m, n) de la imagen de entrada.
- $K(i - m, j - n)$ es el valor del kernel en la posición $(i - m, j - n)$.
- $\sum_m \sum_n$ representa la suma de la acción del *kernel* sobre la salida de la capa anterior.

En el caso de la primera capa convolucional, $Y_{l-1}(m, n) = I(x, y)^{RGB}$. En la Figura 1.4 se muestra el comportamiento de la convolución de manera gráfica.

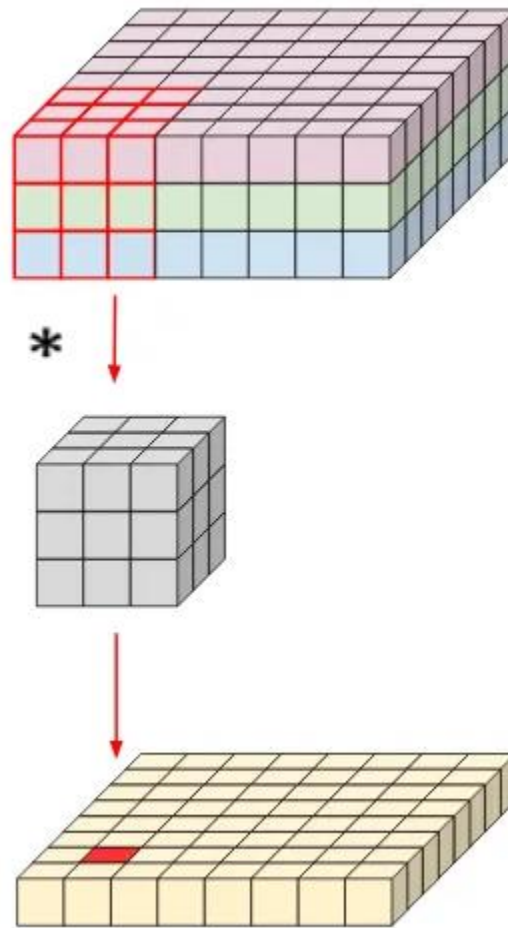


Figura 1.4 Ejemplo de convolución en imagen RGB [20].

1.5.2 ReLU

Una capa de función de activación es una parte fundamental en una red neuronal que introduce la no linealidad en el modelo. Se aplica a la salida de una capa previa y permite que la red neuronal aprenda relaciones y patrones complejos en los datos. Una función de activación comúnmente utilizada es la función de activación *Rectified Linear Unit* (*ReLU*). La función *ReLU* se define matemáticamente como:

$$ReLU(x) = \max(0, x) \quad (1.2)$$

Esta función toma un valor de entrada " x " y devuelve " x " si " x " es mayor o igual a cero, y devuelve cero en caso contrario. En otras palabras, la función *ReLU* mapea cualquier valor negativo a cero, manteniendo los valores positivos sin cambios.

1.5.3 Leaky- ReLU

La función de activación *Leaky ReLU* se introdujo para abordar un problema que puede surgir con la función *ReLU*, donde los valores negativos se hacen cero. La función *ReLU* tiene una limitación conocida como el problema de "neuronas muertas" (*dying ReLU problem*). Este problema ocurre cuando los pesos tienen valores negativos, lo que resulta en que la neurona nunca se activa (su salida siempre es cero). Cuando esto sucede, la neurona no aprende y se vuelve inútil para la red.

La función *Leaky ReLU* se define como:

$$Leaky\ ReLU(x) = \begin{cases} \alpha * x, & \text{para } (x < 0) \\ x, & \text{para } (x \geq 0) \end{cases} \quad (1.3)$$

Si " x " es mayor o igual a cero, la función *Leaky ReLU* devuelve " x ", al igual que la *ReLU*. Sin embargo, si " x " es negativo, *Leaky ReLU* devuelve $\alpha * x$, donde $\alpha < 1$. Esta pendiente pequeña permite un flujo de información débil a través de las neuronas inactivas, lo que evita que se vuelvan completamente inactivas y ayuda a resolver el problema de las "neuronas muertas".

1.5.4 Capa de reducción de muestreo (*Pooling*)

La capa de reducción de muestreo, como el max-pooling, reduce el ruido en los mapas de características. Esto se asemeja a cómo la corteza visual humana simplifica la información, centrándose en lo relevante y omitiendo lo innecesario para una percepción y procesamiento eficaces. De manera análoga, el max-pooling y técnicas similares identifican características clave, eliminando ruido y permitiendo que las redes neuronales extraigan patrones significativos [21]. La ecuación de *pooling* [22] en el contexto de las imágenes en una FCN se utiliza para reducir la resolución espacial de los mapas de características de la siguiente manera:

$$Y(i, j, c) = Pooling(X(i, j, c)) \quad (1.4)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar la operación de *pooling* en la posición (i, j, c) del mapa de características de salida, en donde (i, j) son las coordenadas espaciales que indican la posición en la imagen y (c) representa el canal de color de la imagen.

- $X(i, j, c)$ es el valor en la posición (i, j, c) del mapa de características de entrada.

1.5.5 Max-pooling

El *max-pooling* es una función *pooling* que selecciona el valor máximo dentro de una ventana del mapa de características de entrada. Esta función se da como:

$$Pooling(X(i, j, c)) = \max_{(m, n)} X(i \cdot s + m, j \cdot s + n, c) \quad (1.5)$$

donde (m, n) denota las posiciones dentro de la región de *pooling* y s es el factor de *stride*, que especifica la separación entre las ubicaciones de *pooling* [23].

1.5.6 Upsampling

El *Upsampling* o aumento de tamaño es un procedimiento que agrega datos en un tensor. Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar la operación de *Upsampling* en la posición (i, j, c) del mapa de características de salida.
- $X(i, j, c)$ es el valor en la posición (i, j, c) del mapa de características de entrada.

En *Upsampling*, se repiten los valores de los píxeles en la dirección espacial para aumentar la resolución de la siguiente manera:

$$Upsampling(X(i, j, c)) = X\left(\frac{i}{s}, \frac{j}{s}, c\right) \quad (1.6)$$

donde s es el factor de aumento de tamaño, que especifica cuántas veces se duplica el tamaño en cada dimensión.

Este procedimiento evita que el modelo se sesgue hacia la clase mayoritaria y se define como:

$$Y(i, j, c) = Upsampling(X(i, j, c)) \quad (1.7)$$

1.5.7 Descripción de funcionamiento del *autoencoder*

Un *autoencoder* es una red neuronal que busca aprender una representación codificada (latente) de los datos de entrada mediante la reducción de dimensiones y, posteriormente, la reconstrucción de los datos originales desde la representación latente [24]. El *autoencoder* tiene tres etapas: codificador, cuello de botella y decodificador. El codificador es la parte de la red

que realiza la transformación lineal y no lineal para obtener la representación latente. Las capas del codificador se describen como: Sea T el conjunto de datos de entrada, y z un vector de representación latente. El codificador aplica una transformación lineal a la entrada T utilizando una matriz de pesos W_1 y un sesgo b_1 para obtener una representación intermedia h :

Función lineal

$$h = W_1 \bullet T + b_1 \quad (1.8)$$

donde $\bullet$ representa la operación de multiplicación matricial.

Luego, se aplica una función de activación no lineal g_1 a la representación intermedia h para obtener la representación latente z :

Función de activación

$$z = g_1(h) \quad (1.9)$$

La función de activación g_1 introduce no linealidad a la representación latente, lo que permite que el *autoencoder* capture relaciones no lineales en los datos de entrada.

La ecuación que describe el cuello de botella es la representación latente (z) que se obtiene después de aplicar la transformación lineal y la función de activación en el codificador:

$$z = g_1(W_1 \bullet T + b_1) \quad (1.10)$$

La ecuación que describe el decodificador se puede expresar matemáticamente de la siguiente manera: sea z la representación latente y $\hat{x}$ la reconstrucción de los datos de entrada X . El decodificador aplica una transformación lineal a la representación latente z utilizando una matriz de pesos W_2 y un sesgo b_2 para obtener una representación intermedia h' :

Función lineal

$$h' = W_2 \bullet z + b_2 \quad (1.11)$$

donde $\bullet$ representa la operación de multiplicación matricial. Luego, se aplica una función de activación no lineal g_2 a la representación intermedia h' para obtener la reconstrucción $\hat{x}$ de los datos de entrada T :

Función de activación

$$\hat{x} = g_2(h'). \quad (1.12)$$

La función de activación g_2 introduce no linealidad a la representación intermedia h' , lo que permite que el autoencoder realice la reconstrucción de los datos originales con precisión.

1.6 Redes seleccionadas para el análisis

Se implementaron 3 modelos del estado del arte, U-Net debido al uso de convoluciones aunado a su estructura de autoencoder generando resultados que en el área médica en específico de segmentación han sido sumamente sorprendentes [25], MobileNet debido al uso de convoluciones y a su diseño pensado en usarse en dispositivos móviles [26] y por ultimo ResNet que al igual que MobileNet utiliza convoluciones para generar resultados enfocándose en detalles que otros modelos tienen dificultades de generar, estos resultados se generan debido a la técnica de “capas residuales” [27].

Las redes ResNet y MobileNet se implementaron con una capa de entrada de 500x500 con 3 canales (RGB) y la salida se adaptó a 500x500 con 1 canal (escala de grises). U-net se implementó con una entrada en 496x496 con 3 canales debido a como está estructurado internamente, la salida también se dejó en 496x496 con 1 canal. A continuación, ResNet, MobileNet y U-Net se describirán brevemente.

1.6.1 ResNet

ResNet fue propuesta en el año 2015 e introduce el concepto de “aprendizaje residual” [27] en redes convolucionales. Las redes neuronales residuales se realizan con saltos de 2 a 3 capas y no existe linealidad entre ellas, como se observa en la Figura 1.5, donde la salida de un bloque que se conecta a la entrada de los demás bloques que conforman la red neuronal [27].

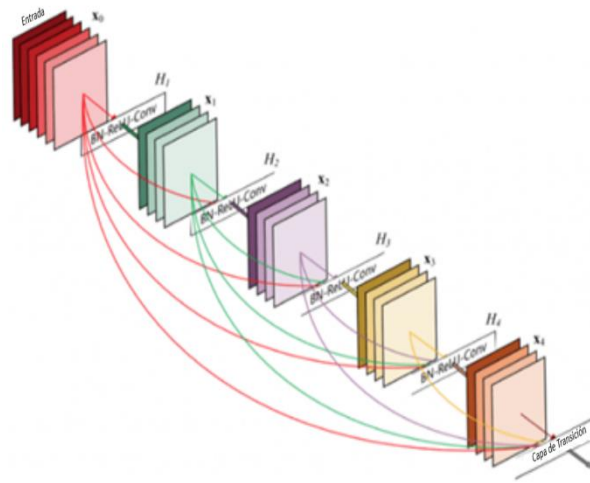


Figura 1.5 Modelo ResNet.

1.6.2 U-Net

U-Net es una red neuronal convolucional que se desarrolló para la segmentación de imágenes biomédicas en el Departamento de Ciencias de la Computación de la Universidad de Friburgo [25]. La red es una FCN cuya arquitectura está especializada en entrenar con pocas imágenes, pero sin disminuir el desempeño en la segmentación. La Figura 1.6 muestra la arquitectura de la U-Net.

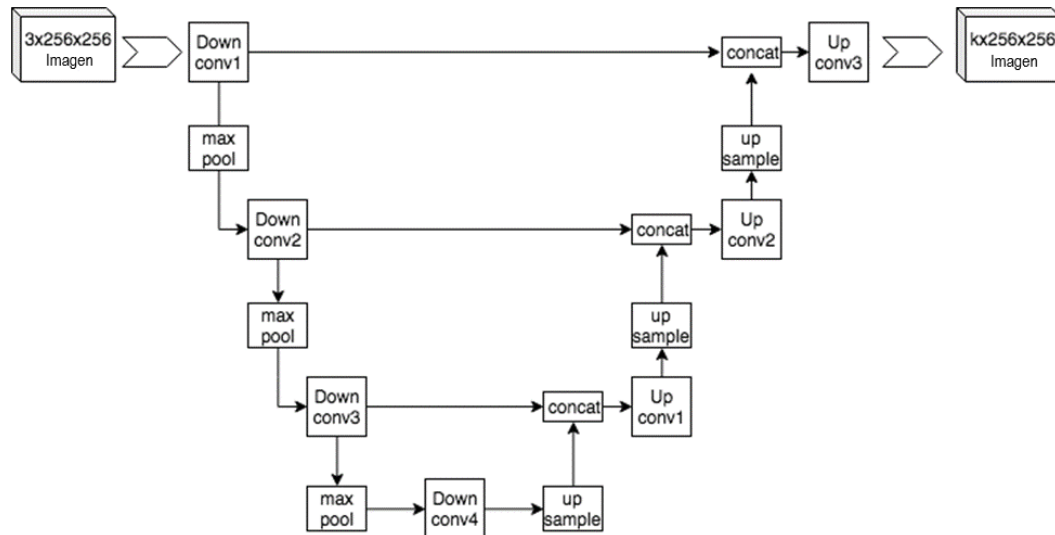


Figura 1.6 Modelo U-Net.

1.6.3 MobileNet

Es un modelo de red neuronal con bajo costo computacional, baja latencia, bajo consumo de energía y parametrizado para su uso en sistemas embebidos y móviles [26]. Se puede utilizar para clasificación, detección, segmentación, etc.

MobileNet compensa entre su relación latencia, tamaño y precisión comparándose con otros modelos conocidos del estado del arte. En la Figura 1.7 se puede apreciar la estructura del modelo.

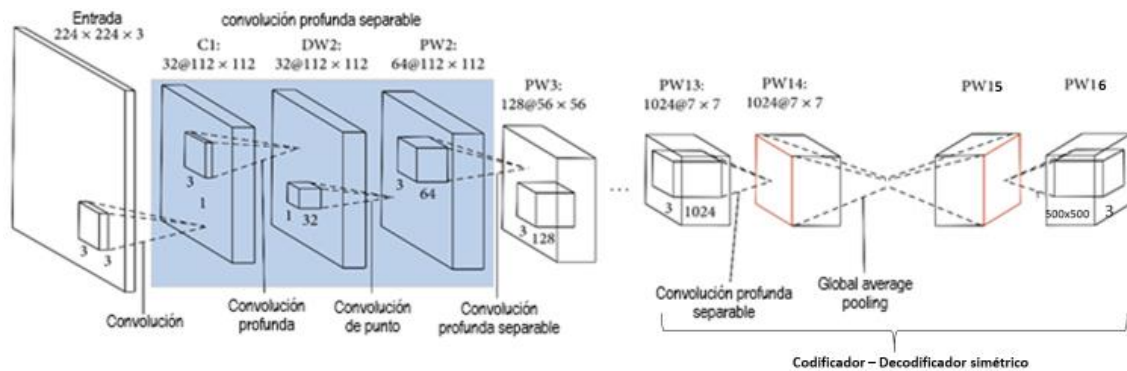


Figura 1.7 Modelo MobileNet.

2 BASE DE DATOS

Esta investigación dio origen a una base de datos llamada “Imágenes reales de bacterias y levaduras” o “*Bacteria and Yeast Real Images*” (BYRI). BYRI cuenta con 3 clases distintas, las cuales son:

- 496 imágenes de *Candida albicans*.
- 500 imágenes de *Escherichia coli*.
- 506 imágenes de *Staphylococcus aureus*.

Este capítulo se presenta de la siguiente manera. La sección 2.1 presenta la base de datos y la sección 2.2 presenta la composición de las imágenes de la base de datos. Finalmente, la sección 2.3 presenta el aumento de datos.

2.1 Bases de datos en análisis clínico

Las imágenes de BYRI fueron generadas en la Facultad de Medicina y Ciencias Biomédicas de la UACH. Personal de la UACH consideró importante la elaboración de esta base de datos ya que en el análisis de literatura realizado no se reportaron *datasets* de microorganismos biomarcadores de patologías.

El desarrollo de los *ground truth* involucró una residencia profesional, dos servicios sociales y la presente tesis. En la residencia profesional y los servicios sociales se realizaron las regiones que representan las bacterias y levaduras presentes en las muestras realizadas por la UACH. Sin embargo, en este trabajo se tomaron en consideración las especificaciones de especialistas de la UACH para rediseñar dichas regiones y formar los *ground truths*. Dichas especificaciones fueron:

- Para el caso de CA se detallaron los criterios que permiten distinguir su forma característica de levadura con brotes, asegurando que cada célula o agrupación de células se anotara adecuadamente. Además, se especificaron las variaciones posibles en la morfología de este hongo, como su transición a una forma filamentosa cuando se encuentra en condiciones de estrés.
- En el caso de EC el especialista proporcionó indicaciones precisas sobre cómo identificar las células bacterianas individuales, considerando su forma de bastón o

bacilo, y teniendo en cuenta la posibilidad de que se encuentren en colonias o agrupaciones.

- Asimismo, para SA, se indicaron los atributos distintivos que permiten diferenciarlo de otros microorganismos. Se enfatizó la importancia de identificar su característica disposición en racimos de uvas, lo que lo distingue de otros tipos de bacterias que suelen presentar otras formas de agrupación.

El especialista también enfocó la atención en la correcta identificación de las áreas de interés, ya que los microorganismos pueden encontrarse en entornos complejos con diversos elementos celulares y detritos. Por tanto, se proporcionaron pautas para evitar confusiones y mejorar la precisión de la segmentación.

El proceso de anotación de los *ground truth*, siguiendo estas especificaciones, fue meticuloso y riguroso. Se realizaron múltiples revisiones para asegurar la coherencia y calidad de los datos etiquetados, lo que permitió construir un conjunto de datos confiable y representativo de los microorganismos de interés, como se muestra en la Figura 2.1

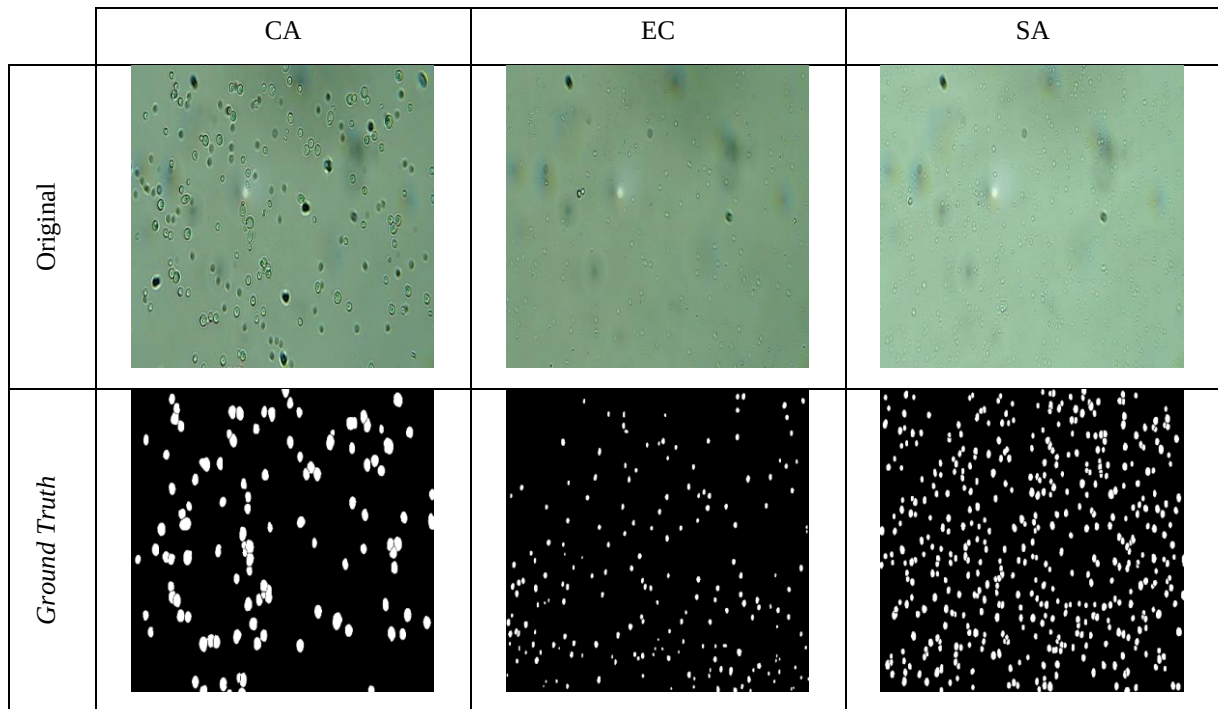


Figura 2.1 Ejemplos de ground truth generados de cada clase

2.2 Descripción de la base de datos

Las imágenes que posee la base de datos BYRI tienen ciertas características que dificultan la identificación de elementos por medio de un procesamiento de imágenes tradicional, algunos de los problemas que tienen estas imágenes son:

- Fuente de luz (foco del microscopio) que genera sobreexposición y contraste excesivo
- Fondo irregular
- Lente de la cámara (y/o microscopio) sucia
- Elementos de interés borrosos
- Oclusión de elementos de interés
- Existencia de artefactos que dificultan la identificación de elementos
- Elementos de interés muy cerca uno de otro
- Dimensiones variadas de las imágenes
- Elementos de interés parecidos al tono del fondo (camuflaje)
- Elevada cantidad de elementos de interés por imagen
- Elementos de interés pequeños

En la Figura 2.2 se muestra un ejemplo de lo mencionado. Se podrían reducir algunos de los inconvenientes en la base de datos como la fuente de luz o lente sucia, lo cual ayudaría en gran medida la calidad de las imágenes. Sin embargo, el fin de esta base de datos generada por la Facultad de Medicina y Ciencias Biomédicas (FMCB) de la Universidad Autónoma de Chihuahua es que la Inteligencia Artificial resuelva la interpretación de los microorganismos con muestras adquiridas en un laboratorio en condiciones reales y cotidianas.

El problema que busca resolver la Facultad de Medicina y Ciencias Biomédicas (FMCB) de la Universidad Autónoma de Chihuahua es realizar el conteo de microorganismos por medio de un algoritmo de computadora, por lo que es necesario generar los *ground truth* que presentan las regiones que representan los elementos de interés para obtener la morfología, ubicación y cantidad de los microorganismos.

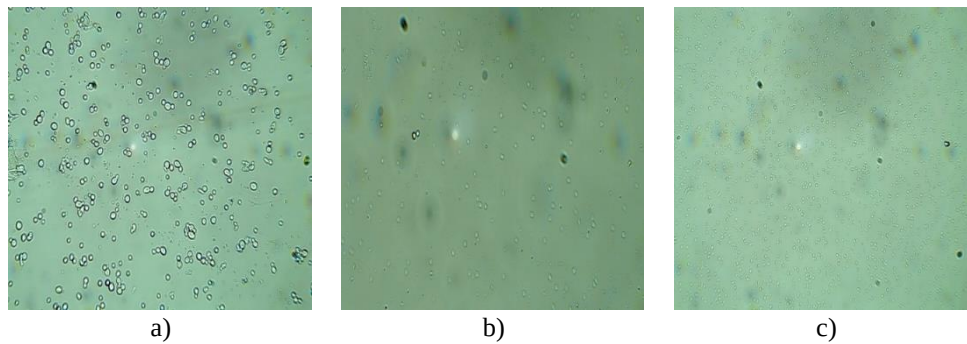


Figura 2.2 Muestra de imágenes de base de datos BYRI
a) *Candida Albicans* b) *Escherichia Coli* c) *Staphylococcus Aureus*.

2.3 Aumento de datos

El método de aprendizaje seleccionado fue el supervisado, debido a que los microorganismos tienen características morfológicas muy específicas, y el fondo tiene ruido debido a objetos como foco del microscopio, lente de la cámara sucia, elementos de interés parecidos al tono del fondo (camuflaje), etc. Mediante el aprendizaje supervisado, se proporcionó al algoritmo un conjunto de datos etiquetados, donde cada muestra estaba asociada con la clase correspondiente (microorganismo o fondo). De esta manera, M-Net debe aprender a reconocer y distinguir las características relevantes de los microorganismos y separarlas de las características de los objetos de fondo.

Para lograr este nivel de aprendizaje, fue necesario realizar un aumento de datos que consiste en aplicar técnicas de transformación a las imágenes existentes para incrementar el tamaño del conjunto de datos [28]. En este caso, se utilizaron técnicas de rotación de las imágenes en los ejes de las abscisas “x” (ecuación (2.1)), “y” (ecuación (2.2)) y “xy” (ecuación (2.3)). Estas técnicas son consideradas como métodos de aumento de datos tradicionales y son ampliamente utilizadas en el procesamiento de imágenes. Se tomaron 200 imágenes de cada clase (recordando que cada clase cuenta con aproximadamente 500 imágenes) y se regeneraron las máscaras para obtener el *ground truth* (GT) usando el software *Sketchbook* para editar imágenes en una Tablet. Se fueron remarcando punto a punto los elementos de interés, y los resultados se muestran en la Figura 2.3. En total, se obtuvieron 600 imágenes, de las cuales se extrajeron los datos de la siguiente manera:

- *Candida albicans* → de las 496 imágenes, se tomaron 200 para generar los *ground truth*, de esta manera se tienen 296 imágenes que se emplearon para el *set* de prueba.
- *Escherichia coli* → de las 500 imágenes se tomaron 200 para generar los *ground truth*, de esta manera se tienen 300 imágenes que se emplearon para el *set* de prueba.
- *Staphylococcus aureus* → de las 506 imágenes se tomaron 200 para generar los *ground truth*, de esta manera se tienen 306 imágenes que se emplearon para el *set* de prueba.

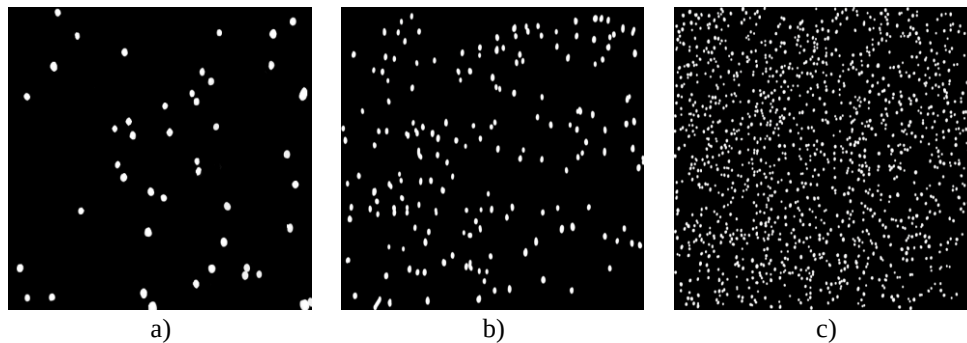


Figura 2.3 Muestras de GT generadas (no están relacionadas a la Figura 2.2).
a) *Candida Albicans* b) *Escherichia Coli* c) *Staphylococcus Aureus*.

La rotación de las imágenes en los ejes de las abscisas “x” e “y” [29] permite que el modelo aprenda a reconocer los microorganismos desde diferentes perspectivas, lo que mejora la capacidad de generalización y la robustez del modelo frente a variaciones de orientación presentes en las imágenes reales [28]. Por otro lado, la rotación en el eje de las abscisas “xy” es útil para simular transformaciones de giro en el plano tridimensional, lo que ayuda al modelo a adaptarse a posibles desalineaciones entre las muestras de entrenamiento y las muestras de prueba [28]. Estas técnicas de aumento de datos permiten que la FCN aprenda patrones invariantes en el espacio de manera más efectiva y evita el sobre entrenamiento. Además, al aumentar la diversidad de datos, se logra una mejor generalización del modelo, lo que se traduce en una mayor precisión en la clasificación y detección de los microorganismos de interés de las imágenes [28].

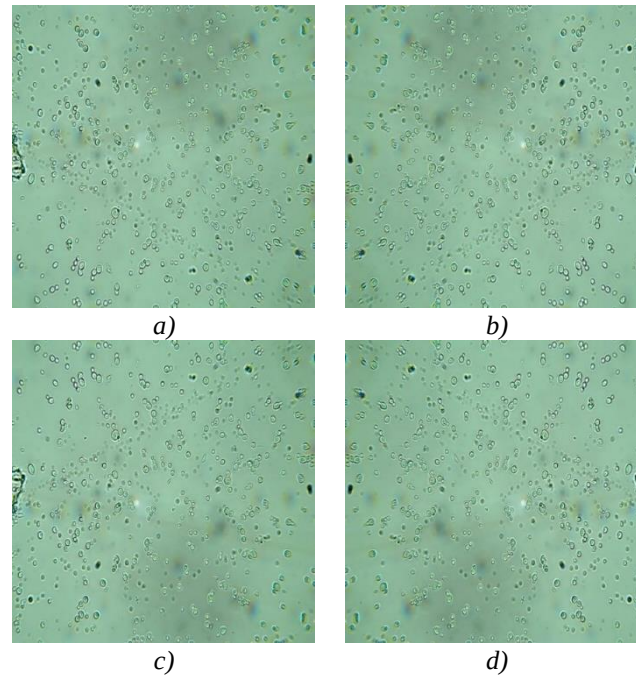


Figura 2.4 Ejemplo de aumento de datos
a) imagen original, b) rotación de 180° eje 'x', c) inversión en eje 'y', d) rotación de 180° e inversión en eje 'y' y en eje 'x'.

2.3.1 Inversión de la imagen

Para explicar la rotación de imágenes y las inversiones en los ejes de las abscisas (“x”, “y” y “xy”), se mostrarán en las siguientes secciones:

2.3.2 Inversión en el eje de la abscisa “x”

La inversión en el eje de la abscisa “x” implica reflejar la imagen horizontalmente. La ecuación para invertir una imagen en el eje de la abscisa “x” es:

$$Y(i, j, c) = X(\text{anch} - i - 1, j, c) \quad (2.1)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar la inversión en el eje x en la posición (i, j, c) de la imagen de salida.
- X es el valor en la posición (i, j, c) de la imagen de entrada.
- anch es el ancho de la imagen.

2.3.3 Inversión en el eje de la abscisa “y”

La inversión en el eje de la abscisa “y” implica reflejar la imagen verticalmente. La ecuación para invertir una imagen en el eje de la abscisa “y” es:

$$Y(i, j, c) = X(i, alt - j - 1, c) \quad (2.2)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar la inversión en el eje y en la posición (i, j, c) de la imagen de salida.
- X es el valor en la posición (i, j, c) de la imagen de entrada.
- alt es la altura de la imagen.

2.3.4 Inversión en el eje de la abscisa “xy”

La inversión en el eje de la abscisa “xy” implica reflejar la imagen tanto horizontal como verticalmente. La ecuación para invertir una imagen en el eje de la abscisa “xy” es:

$$Y(i, j, c) = X(anch - i - 1, alt - j - 1, c) \quad (2.3)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar la inversión en ambos ejes en la posición (i, j, c) de la imagen de salida.
- X es el valor en la posición (i, j, c) de la imagen de entrada.
- $anch$ es el ancho de la imagen.
- alt es la altura de la imagen.

3 MODELO PROPUESTO

Este capítulo presenta las FCNs propuesta en esta tesis, las cuales se diseñaron como redes neuronales *autoencoders* con baja cantidad de parámetros y su arquitectura se basa en convoluciones, convoluciones profundas y convoluciones profundas separables. Las FCN se denominaron Red de Conteo de Microorganismos (M-Net). La metodología de diseño de los modelos propuestos se basó en la unión entre modelos tipo FCN y *Autoencoders*.

Las M-Net propuestas son un trabajo original de esta tesis, pero tienen similitudes con la red U-Net, ya que parte de U-Net comprime la información demasiadas veces y realiza varias concatenaciones que no necesariamente aportan información relevante. Sin embargo, M-Net toma la información principal de la imagen y la pasa por un *autoencoder* que comprime la imagen 2 veces.

Para describir las redes diseñadas, el presente capítulo se organiza de la siguiente manera: la sección 3.1 describe al modelo M-Net V1, así como el funcionamiento de sus bloques principales. La sección 3.2 expone las mejoras de la versión anterior, así como una descripción de sus bloques. La sección 3.3 detalla el progreso que se obtuvo en la versión final M-Net V3 junto con una explicación de sus cuatro bloques. Finalmente, la sección 3.4 presenta la función de pérdida utilizada para en entrenamiento.

3.1 M-Net V1

Es una FCN con una arquitectura de *autoencoder* simétrico [30]. El modelo se determinó basándose en la estructura de los *autoencoder*, además de ir viendo los resultados que se generaban y el mapa de características con la intención de aprovechar lo mejor posible los datos suministrados, el modelo se compone de tres etapas (como se aprecia en la Figura 3.1) con un bajo número de capas (se pueden contar en la Figura 3.2 y/o Figura 3.3). Esto significa que va a tener la misma cantidad de capas en el codificador y en el decodificador, tal como se muestra en la Figura 3.3. La arquitectura de M-Net V1 se distribuye en pares de capas convolucionales, y se tiene 4 tipos de configuraciones con tamaños distintos, tal como se muestra en la Figura 3.2 y Figura 3.3 (el color de fondo de los bloques de la Figura 3.3 corresponde a los bloques de la Figura 3.1).

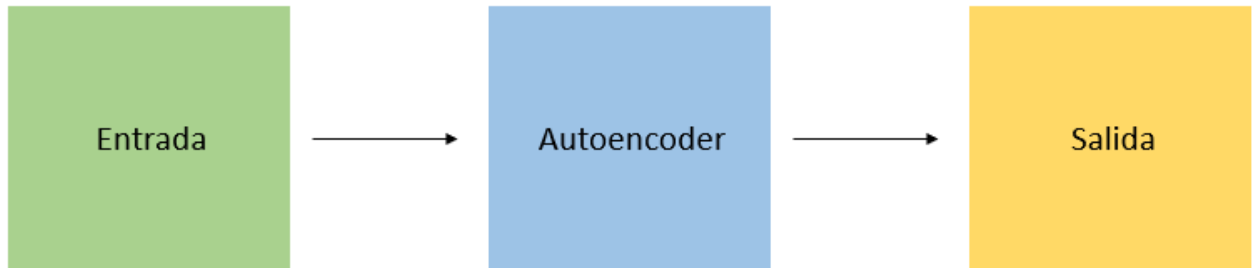


Figura 3.1 Representación sintetizada de M-Net V1 y V2.

M-Net V1

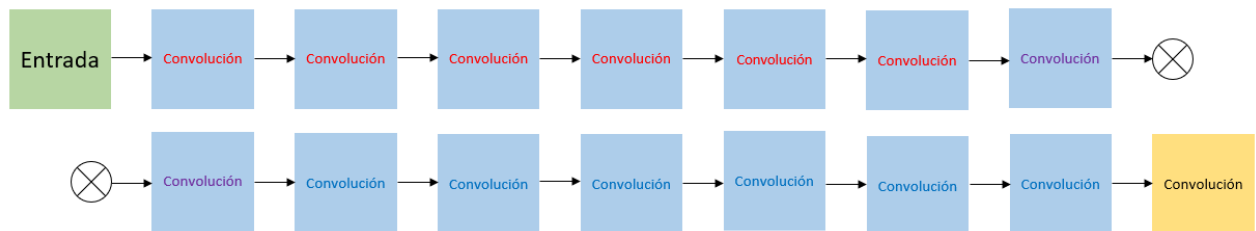


Figura 3.2 Convoluciones de M-Net V1.

Arquitectura del Modelo M-Net v1

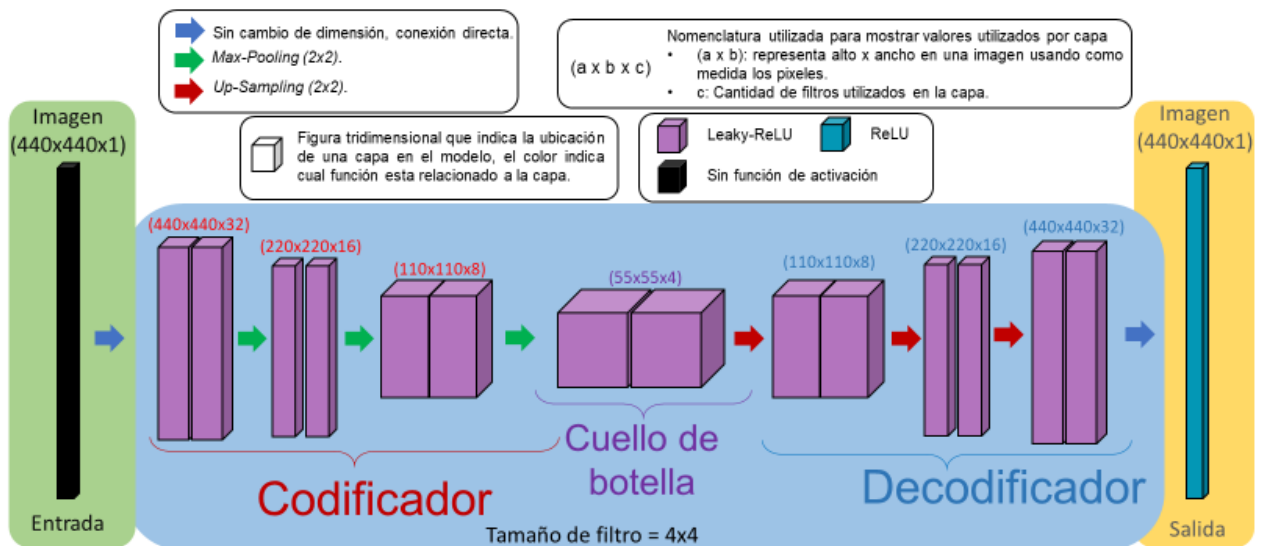


Figura 3.3 Arquitectura del modelo de autoencoder diseñado (M-Net V1).

De esta manera, la arquitectura de M-Net V1 es la siguiente:

Entrada: recibe los datos que entran al modelo, y se refiere a una imagen $I(x, y)$ con resolución espacial de 440x440 a 8 bits y escala de grises (440x440x1). La resolución espacial fue propuesta a 440x440 debido a que fue necesario estandarizar el tamaño de las imágenes y las capacidades de cómputo. El uso de un canal fue por reducir el costo computacional.

Red Totalmente Convolutiva (FCN): se encarga de comprimir la información de la imagen para desprejar la información irrelevante del fondo y segmentar los microorganismos. Todas las capas de la FCN en el modelo M-Net V1 están definidas por convoluciones dadas por la ecuación (1.1) con función de activación *Leaky ReLU* dada por la ecuación (1.3). El uso de la *Leaky ReLU* es para que los valores negativos junto con las operaciones de convolución, se pueden extraer bordes, identificar patrones morfológicos y eliminar características no relevantes [31]. Los bloques de M-Net V1 tienen la siguiente estructura:

- *Codificador:* de acuerdo con la Figura 3.3, tiene tres pares de capas con función *max-pooling* definida por la ecuación (1.5) con ventanas de 2x2, en donde, las dimensiones de los pares del codificador son 440x440x32, 220x220x16 y 110x110x8, los valores mencionados anteriormente se definieron para tener una reducción progresiva de dimensiones a través de las capas de *max-pooling*, con este objetivo, los tres pares de capas se encargan de comprimir la información en el espacio latente con el fin de mantener los microorganismos de interés y reducir/eliminar toda la información que sea innecesaria de la imagen que no contribuye al proceso de segmentación. Este enfoque de compresión selectiva garantiza que el modelo pueda centrarse en los aspectos relevantes de la tarea.
- *Cuello de botella:* tiene un par de capas convolucionales con una dimensión de 55x55x4 para generar una etapa final de compresión que mejora aún más la representación de los microorganismos y sus características clave, preparando el terreno para la siguiente etapa de decodificación y reconstrucción. El cuello de botella obtiene los mapas de características abstractas que estimulan los píxeles que pertenecen a los microorganismos de interés e inhiben lo demás. Dado a que M-Net V1 es simétrico, la segunda capa del cuello de botella tiene un *up-sampling* definido por la ecuación (1.7) con ventanas de 2x2.

- *Decodificador*: tiene tres pares de capas con dimensiones de 110x110x4, 220x220x16 y 440x440x32 conectadas por un *up-sampling* de 2x2. Estas capas desempeñan un papel fundamental en la reconstrucción de la imagen segmentada a partir de la representación comprimida del espacio latente. El proceso de decodificación busca expandir gradualmente las dimensiones mientras se recupera la información esencial perdida durante la etapa de codificación.

Salida: es una capa con una función de activación *ReLU* definida por la ecuación (1.2) para que las características definidas por M-Net V1 tomen valores positivos. Mediante la función *ReLU*, se genera una salida no lineal que activa las características definidas por la (FCN), asegurando que tomen valores positivos. Esta función de activación *ReLU*, tiene la propiedad de permitir el paso de valores positivos sin modificarlos, mientras que anula los valores negativos, promoviendo así la activación de las características más relevantes y resaltando las señales de interés en los datos procesados por la red. Los resultados que arroja M-Net V1 es un mapa de características que inhibe el fondo con valores de niveles de grises cercanos a cero y estimula los microorganismos con valores de pixel cercanos a 255 (ver Figura 3.4). Por ello, el siguiente paso en la salida es segmentar el resultado de M-Net V1 mediante el método OTSU [32]. De esta manera, se generan *blobs* que representan los microorganismos de interés.

M-Net se entrenó con *backpropagation* [33], el optimizador Adam, tasa de aprendizaje de 0.005, tamaño de lote de 10 y 100 épocas. Estos parámetros de aprendizaje se seleccionaron con base en limitaciones técnicas, y por experimentación (estos parámetros también aplican a M-Net versión 2 y 3 que se describirán en este mismo capítulo).

3.1.1 Muestras de resultados del modelo

La Figura 3.4 muestra los resultados cualitativos obtenidos por el modelo y de acuerdo con ellos, se interpreta lo siguiente:

- **CA.** Se obtuvo una morfología adecuada, pero tiene problemas al asignar valores altos a los pixeles de los elementos de interés (los elementos se ven grises y no blancos)
- **EC.** La morfología se considera con resultados intermedios, y se limita a asignar puntos pequeños a los microorganismos, al igual que CA los valores de pixel asignados no son altos.

- **SA.** La morfología resultante fue insuficiente ya que los microorganismos no son correctamente representados, en el mejor de los casos se muestran puntos por cada elemento, la mayoría de los casos se agrupa varios elementos en “líneas”, al igual que en CA y EC se tiene el mismo problema con el nivel de gris en los píxeles.

Este diseño fue el que comenzó a dar frutos y fijó un camino a seguir en la investigación realizada. Sin embargo, se decidió generar otras versiones para mejorar la eficiencia y resultados del modelo.

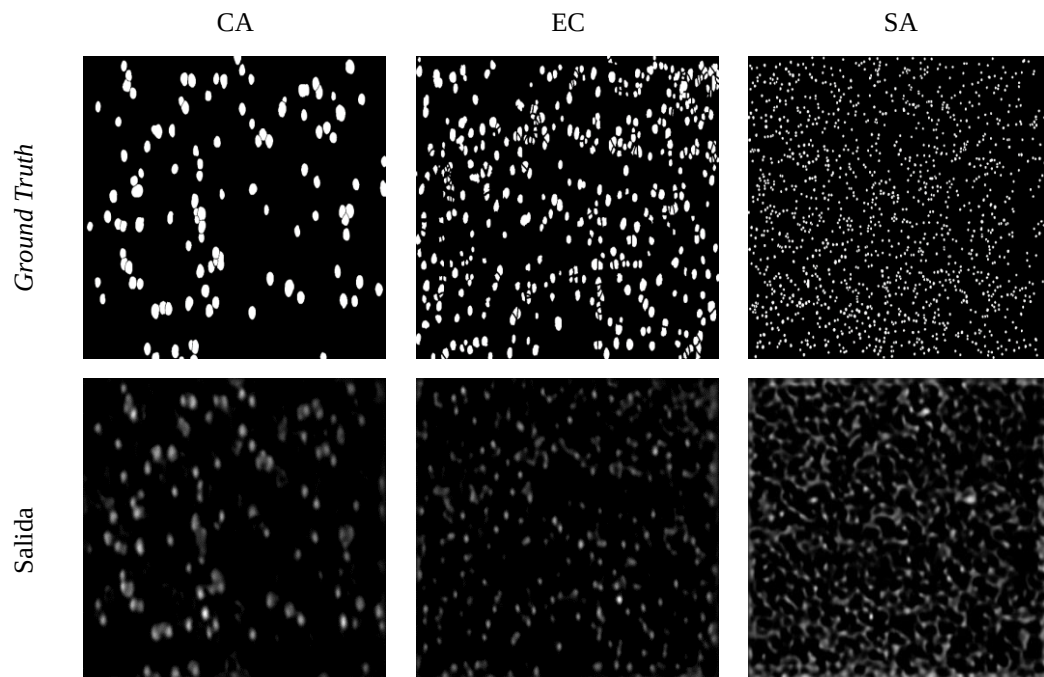


Figura 3.4 Resultados del modelo V1.

3.2 M-Net V2

La versión 2 de M-Net tiene una estructura similar a la Figura 3.1 y se diseñó para reducir el costo computacional y para mejorar el desempeño en la detección de microorganismos. Las principales diferencias de esta versión respecto a la anterior son: se redujo en un par de capas convolucionales (Figura 3.5 y Figura 3.6) debido que al observar los mapas de características se encontró que al reducir la resolución espacial de la imagen en 3 capas le dificultaba al modelo realizar una reconstrucción del resultado y también se cambió la cantidad de *kerneles* en las capas convolucionales por el mismo motivo anterior, tal como se muestra en la Figura 3.6.

M-Net V2

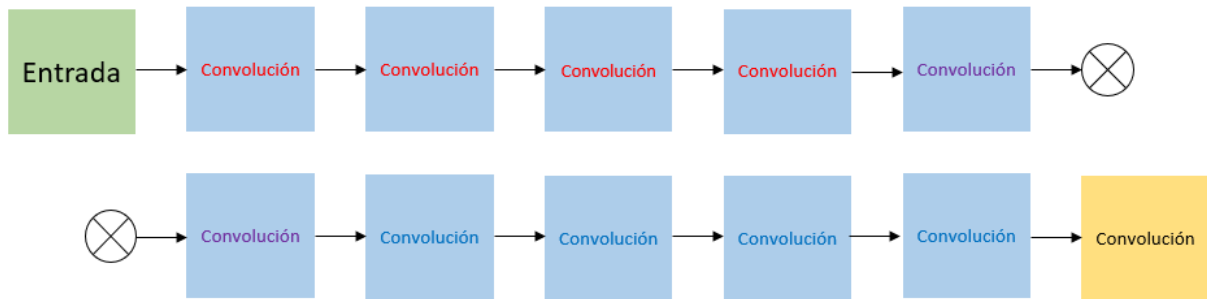


Figura 3.5 Convoluciones utilizadas en M-Net V2.

Arquitectura del Modelo M-Net v2

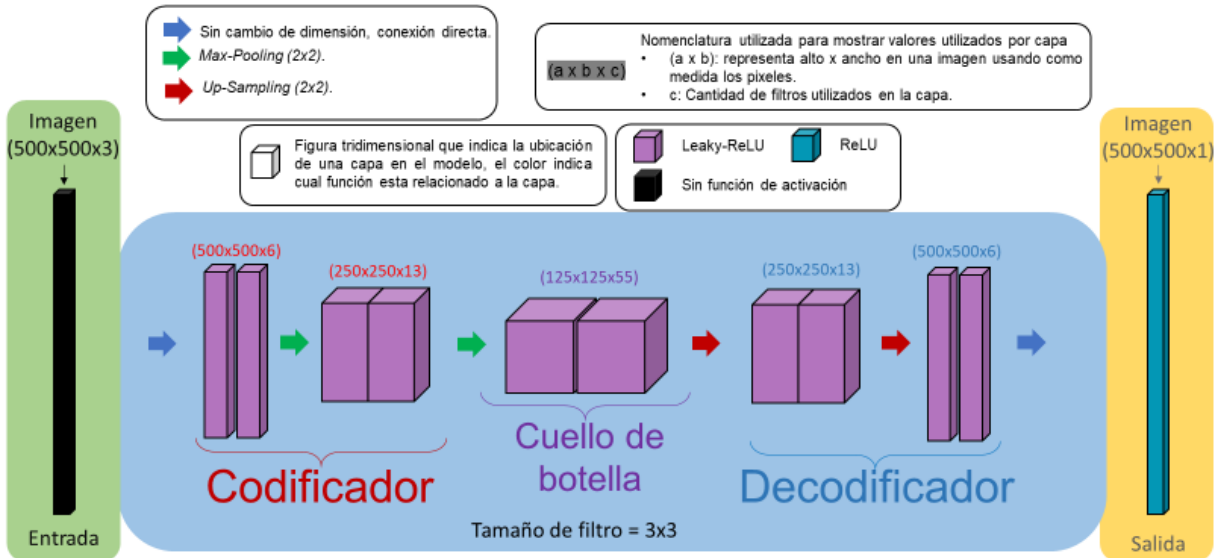


Figura 3.6 Arquitectura del modelo de autoencoder diseñado (M-Net V2).

La arquitectura de M-Net V2 es la siguiente:

Entrada: recibe los datos que entran al modelo, y se refiere a una imagen $I(x, y)$ con una resolución espacial de 500x500 a 8 bits y escala de grises (500x500x1). La resolución espacial fue incrementada ya que en la versión M-Net V1 se agrupaban los blobs de microorganismos que estaban muy unidos.

Red totalmente convolucional: todas las capas de la FCN están definidas por convoluciones dadas por la ecuación (1.1) usando la función de activación *Leaky ReLU* dada por la ecuación

(1.3). El uso de la función de activación *Leaky ReLU* es para que los valores negativos en conjunto con las convoluciones extraen bordes, adquieren patrones morfológicos y reducen características no relevantes [31].

- *Codificador*: de acuerdo con la Figura 3.6, el codificador tiene 2 pares de capas con función *max-pooling* definida por la ecuación (1.5) con ventanas de 2x2. Las dimensiones de los pares de capas del codificador son 500x500x6 y 250x250x13. Estas dimensiones fueron seleccionadas para la reducción progresiva de dimensiones a través de las capas de *max-pooling*. Estos dos pares de capas se encargan de comprimir la información en el espacio latente con el fin de mantener los microorganismos de interés y reducir/eliminar toda la información que sea innecesaria.
- *Cuello de botella*: obtiene los mapas de características abstractas que estimulan los píxeles que pertenecen a los microorganismos de interés e inhiben lo demás. Dado a que M-Net V2 es simétrico, la segunda capa del cuello de botella tiene un *up-sampling* definido por la ecuación (1.7) con ventanas de 2x2.
- *Decodificador*: tiene dos pares de capas con dimensiones de 500x500x6 y 250x250x13 conectadas por un *up-sampling* de 2x2. Estas capas desempeñan un papel fundamental en la reconstrucción de la imagen segmentada a partir de la representación comprimida del espacio latente. El proceso de decodificación busca expandir gradualmente las dimensiones mientras se recupera la información esencial perdida durante la etapa de codificación.

Salida: es una capa con una función de activación *ReLU* definida por la ecuación (1.2) para que las características definidas por la FCN tomen valores positivos. Mediante la función *ReLU*, se genera una salida no lineal que activa las características definidas por la (FCN), asegurando que tomen valores positivos.

3.2.1 Muestras de resultados del modelo

La Figura 3.7 muestra los resultados cualitativos obtenidos por el modelo y de acuerdo con ellos, se interpreta lo siguiente:

CA. La morfología obtenida es adecuada, pero existen problemas al asignar valores altos a los píxeles de los elementos de interés (los elementos se ven grises y no blancos).

EC. La morfología tiene resultados intermedios, ya que se limita a asignar puntos pequeños a los microorganismos, tiene un comportamiento similar a la versión anterior.

SA. El resultado de la morfología es intermedio, si bien esta clase tiene una mejora en comparación a la versión anterior, es común que se agrupen en grandes cantidades debido a la naturaleza de SA, debido a ese comportamiento se tiene cúmulos de píxeles amorfos que no corresponden a los *blobs* esperados por el GT.

Al igual que la versión anterior, el siguiente paso es segmentar el resultado de M-Net V2 mediante el método OTSU. De esta manera, se generan blobs que representan los microorganismos de interés.

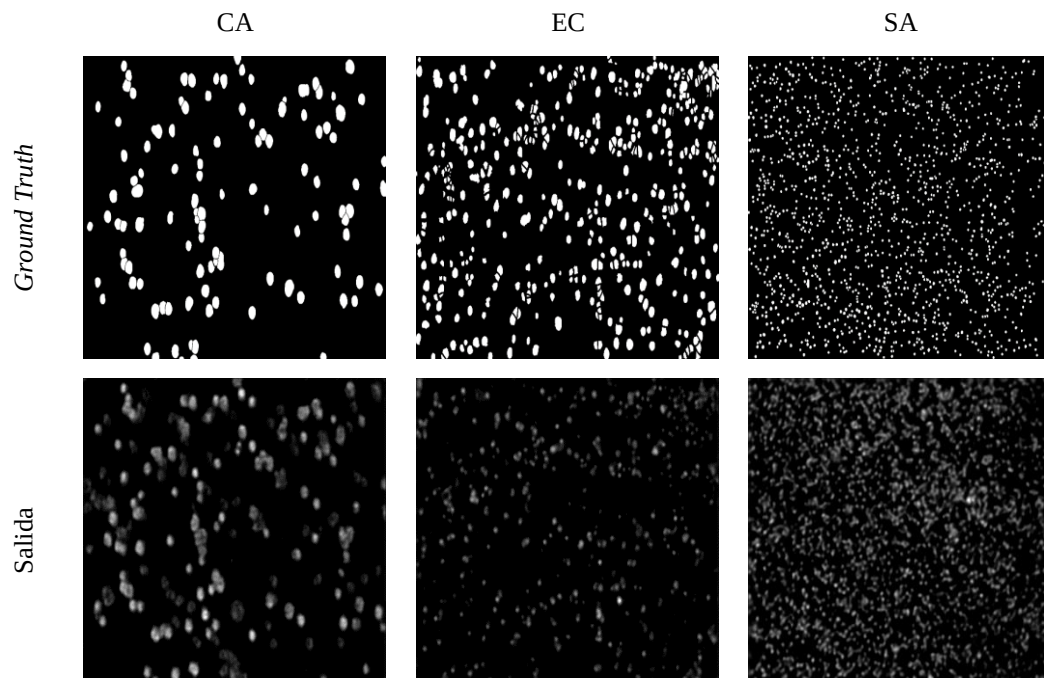


Figura 3.7 Resultados del modelo V2.

3.3 M-Net V3

En esta tercera versión del modelo M-Net, se implementaron ajustes en la disposición de las capas y en la configuración de los *kernel*. Esta decisión se basó en el análisis del mapa de características de la versión previa, así como en el diseño del bloque "reducción de canales", con el objetivo de permitir que el modelo M-Net seleccione de manera efectiva la información relevante contenida en las imágenes RGB. A pesar de mantener el número de capas en el

autoencoder, se modificó el tipo de convoluciones, así como reducir el número de *kernel* usados con la finalidad de reducir la cantidad de parámetros y mejorar la abstracción que tiene el modelo. Como resultado, la versión 3 del modelo logró una reducción del 98% en la cantidad de parámetros en comparación con su predecesora. Estos cambios se llevaron a cabo mediante el uso de convoluciones profundas y convoluciones profundas separables.

3.3.1 Depth-wise convolution

La “*Depth-wise convolution*” [20], que se referirá como "convolución profunda" en este documento de tesis, es una operación que se aplica un *kernel* para cada canal de la capa anterior, la convolución profunda reduce la cantidad de cálculos necesarios en comparación con una convolución tradicional. Esto se debe a que la convolución se aplica de manera independiente a cada canal, evitando así las operaciones de convolución entre canales, además pueden aprender características más específicas en diferentes canales (como bordes, texturas) antes de combinarlas en las convoluciones tradicionales. Esto puede llevar a un aprendizaje más discriminativo.

Las convoluciones profundas están definidas por la siguiente ecuación:

$$Y(i, j, c) = \sum_{(m, n)} X(i \cdot s + m, j \cdot s + n, c) \cdot K(m, n) \quad (3.1)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar una convolución profunda en la posición (i, j, c) del mapa de características de salida.
- X es el valor en la posición (i, j, c) del mapa de características de entrada.
- (m, n) denota las posiciones dentro del filtro (o también llamado kernel) de convolución.
- s es el factor de stride, que especifica la separación entre las ubicaciones de la convolución.
- $K(m, n)$ es el valor del kernel de convolución en la posición (m, n) .

En las convoluciones profundas, se utiliza un conjunto de filtros 2D con dimensiones iguales a la profundidad del tensor de entrada. Durante la operación de convolución, el filtro se multiplica con la región correspondiente del canal y se obtiene una salida llamada "mapa de

características" para cada canal. La Figura 3.8 muestra cómo opera este tipo de convolución [20], donde los mapas de características se combinan mediante concatenación o suma para generar la salida.

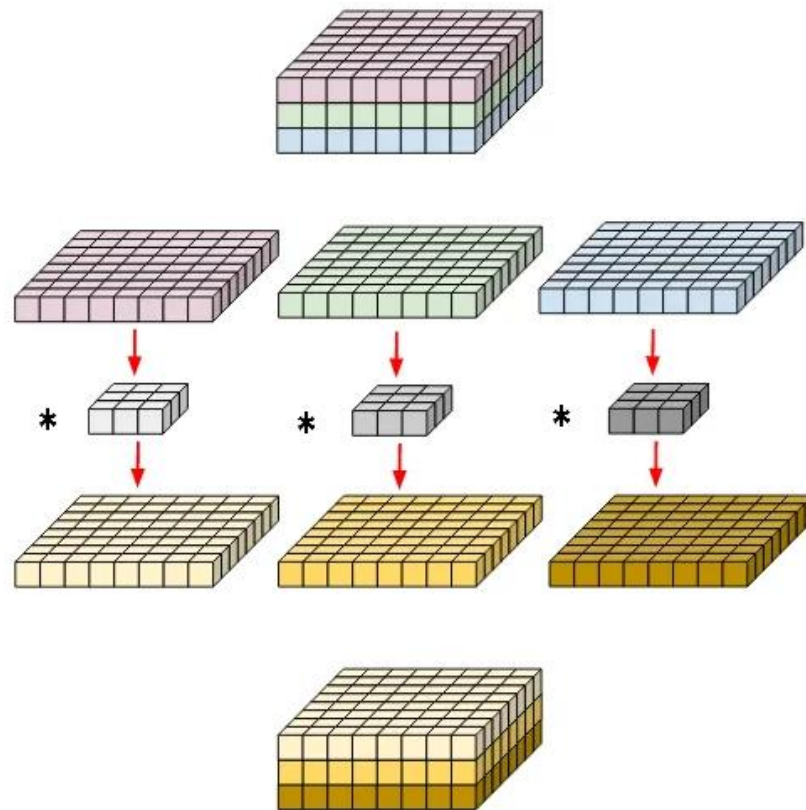


Figura 3.8 Ejemplo de cómo se realiza una convolución profunda en una imagen RGB [20].

3.3.2 Depth-wise Separable Convolution

La *Depth-wise Separable Convolution* [20], que se referirá como "convolución profunda separable" en este documento de tesis, es una técnica utilizada en redes neuronales convolucionales para reducir la cantidad de cálculos y el número de parámetros de entrenamiento. La convolución separable en profundidad se compone de dos etapas: la convolución en profundidad y la convolución de punto. En la etapa de convolución en profundidad, se realiza una convolución separada en cada canal de entrada por con filtros de tamaño reducido para generar un conjunto de mapas de características para cada canal [20].

La convolución de punto realiza una convolución tradicional en los mapas de características generados en la etapa anterior y un *kernel* de tamaño 1x1. De esta manera, las convoluciones separables en profundidad están definidas por la siguiente ecuación:

$$Y(i, j, c) = \sum_{(m,n)} X(i \cdot s + m, j \cdot s + n, c) \cdot K_d(m, n) \cdot K_p(c) \quad (3.2)$$

Donde:

- $Y(i, j, c)$ es el valor resultante después de aplicar las convoluciones profundas separables en la posición (i, j, c) del mapa de características de salida.
- X es el valor en la posición (i, j, c) del mapa de características de entrada.
- (m, n) denota las posiciones dentro del filtro de convolución.
- s es el factor de *stride*, que especifica la separación entre las ubicaciones de la convolución.
- $K_d(m, n)$ es el valor del filtro de convolución profunda en la posición (m, n) , que se aplica a cada canal de entrada por separado.
- $K_p(c)$ es el valor del filtro de la convolución por punto en la posición c , que se aplica a cada píxel y canal de entrada para combinar las características aprendidas.

La convolución separable en profundidad reduce la cantidad de operaciones de multiplicación requeridas en comparación con una convolución tradicional debido a que se utilizan filtros más pequeños en cada etapa y se eliminan las operaciones de convolución entre canales. Como resultado, se reducen los cálculos y se acelera el tiempo de ejecución de la red neuronal [20].

Además, la convolución separable en profundidad también reduce la cantidad de parámetros de entrenamiento en la red. Esto es benéfico en términos de eficiencia de almacenamiento y de capacidad para entrenar redes con un conjunto de datos más pequeño [20]. Además, reducir el número de parámetros disminuye el riesgo de *overfitting* y se mejora la generalización del modelo [20]. Aunque la convolución separable en profundidad puede resultar en una pérdida mínima de precisión en comparación con una convolución tradicional, sigue siendo eficaz en muchas aplicaciones de visión por computadora y aprendizaje profundo.

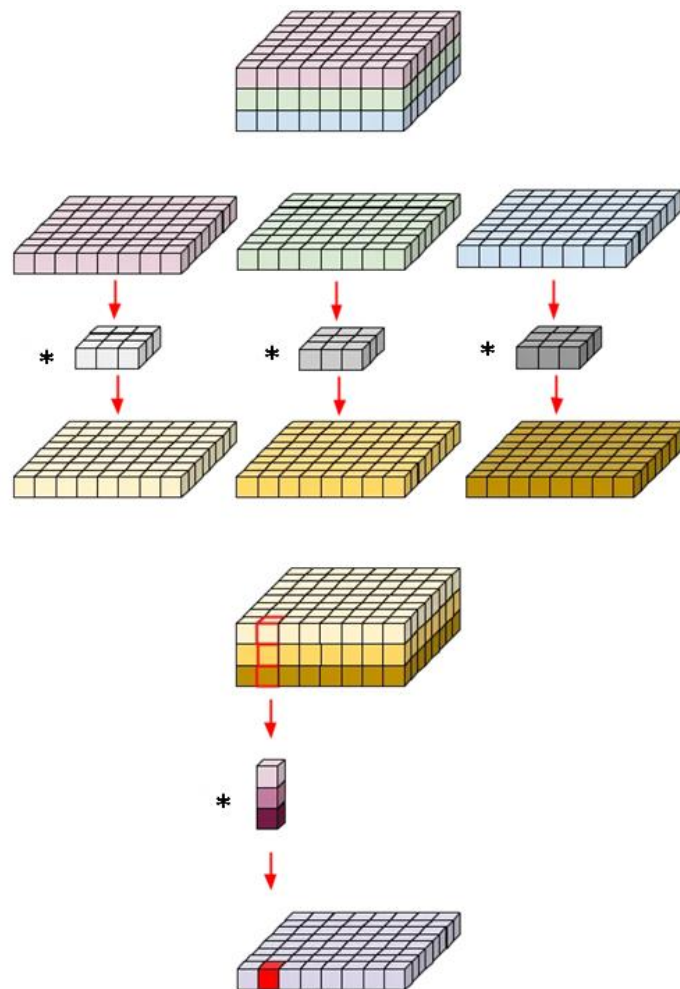


Figura 3.9 Ejemplo de una convolución separable en profundidad en una imagen RGB [20].

3.3.3 Estructura del modelo M-Net V3

En esta versión de M-Net, se utilizaron convoluciones profundas (CP) definidas por la ecuación (3.1), convoluciones profundas separables (CPS) definidas por la ecuación (3.2) y convoluciones (llamadas también convoluciones tradicionales, CT) definida previamente con la ecuación (1.5) para mejorar su desempeño al procesar imágenes en formato RGB. Dichos cambios incluyeron la incorporación de un bloque adicional titulado "Reducción de canales", como se puede apreciar en la Figura 3.10. Sin embargo, cabe mencionar que las versiones 1 y 2 de M-Net originalmente estaban diseñadas para procesar imágenes de un solo canal, lo que inicialmente dificultaba el manejo de imágenes en formato RGB. La adopción de otras técnicas de convolución fue esencial para reestructurar M-Net y optimizar su funcionamiento.

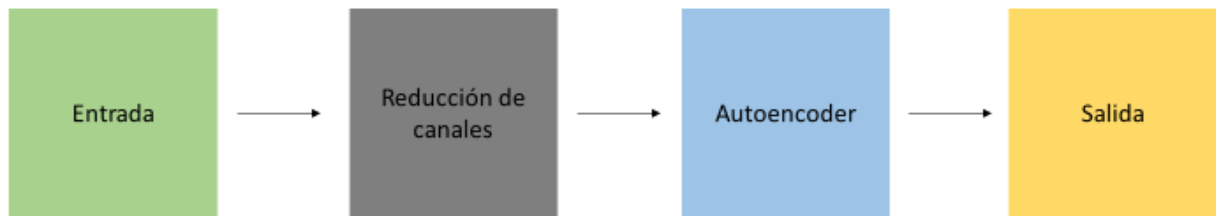


Figura 3.10 Representación sintetizada de M-Net V3.

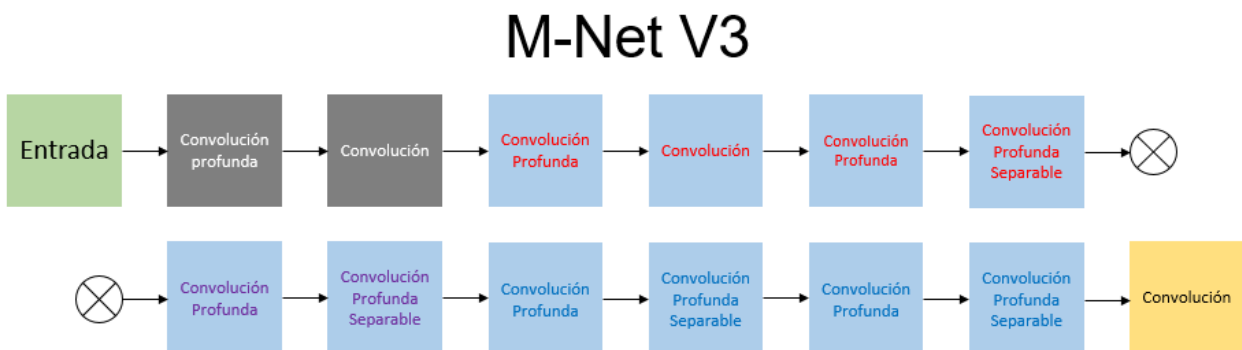


Figura 3.11 Convoluciones utilizadas y orden de acomodo en M-Net V3.

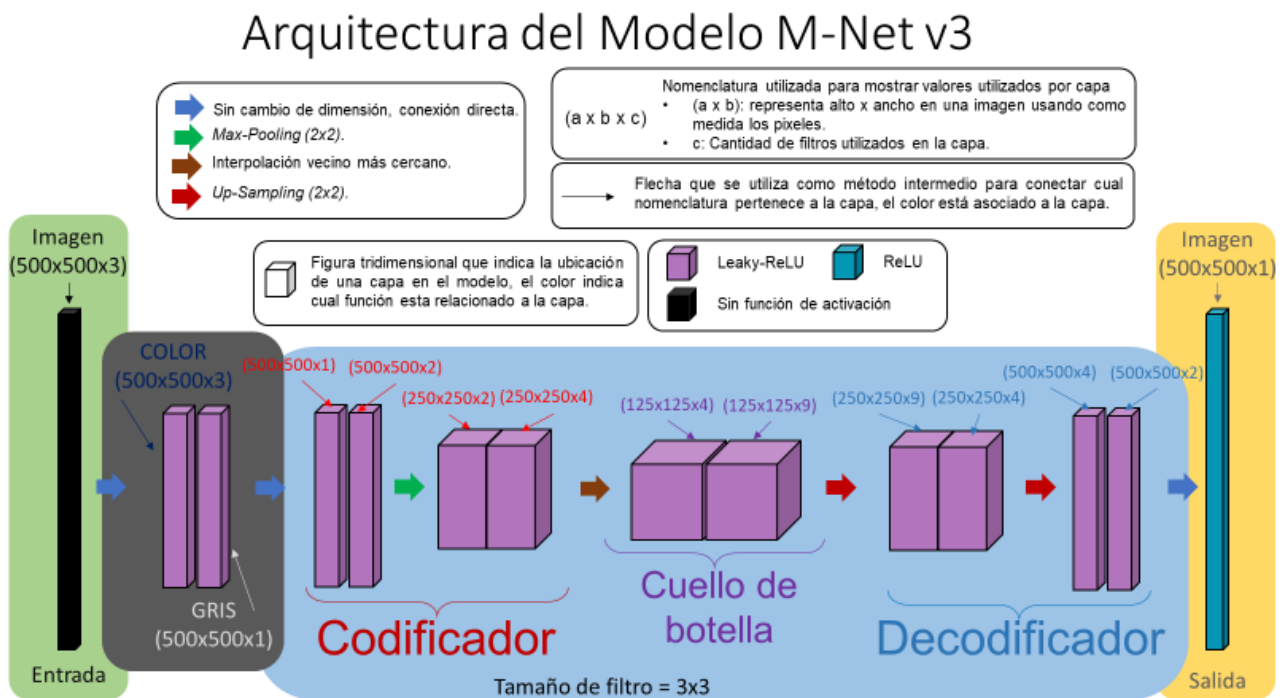


Figura 3.12 Arquitectura del modelo de autoencoder diseñado (M-Net V3).

De esta manera, la arquitectura de M-Net V3 es la siguiente.

Entrada: se refiere a una imagen $I(x, y)^{RGB}$ con una resolución espacial de 500x500 a 8 bits y en formato RGB (500x500x3).

Reducción de canales: este bloque se encarga de generar un componente nuevo basado en la imagen de entrada y una CP por medio de la ecuación (3.1), tal como se observa en la Figura 3.11 y Figura 3.12. Este componente resalta los canales de color relevante y atenúa los no relevantes. Por último, comprime la cantidad de canales y por consiguiente de información, pasando de 3 canales a 1 solo canal por medio de una CT.

Red Totalmente Convolutiva: al igual que las otras versiones, esta FCN tiene 3 etapas, el codificador, el cuello de botella y el decodificador, para todo este bloque se utilizó la función de activación *Leaky ReLU* dada por la ecuación (1.3) para que las convoluciones extraigan los bordes, aprendan patrones morfológicos y reduzcan características que no sean de interés [31].

- *Codificador:* de acuerdo con la Figura 3.12, el codificador en la primera etapa comienza aplicando una CP de 500x500x1 seguida de una CT de 500x500x2 y una función *max-pooling* definida por la ecuación (1.5) con ventanas de 2x2. El segundo par tiene una CP de 250x250x2 seguida de una CPS de 250x250x4. Para esta CPS se tiene un método de interpolación de vecino más cercano [34] debido a que el *max-pooling* entorpece los resultados uniendo características que deberían estar separadas en la imagen original.
- *Cuello de botella:* tiene un par de capas CP-CT con una dimensión de 125x125x4 y 125x125x9 para mejorar la representación de los microorganismos y sus características clave, preparando el terreno para la siguiente etapa de decodificación y reconstrucción. El cuello de botella obtiene los mapas de características abstractas que estimulan los píxeles que pertenecen a los microorganismos de interés e inhiben lo demás. La segunda capa del cuello de botella tiene un *up-sampling* definido por la ecuación (1.7) con ventanas de 2x2.
- *Decodificador:* dado que M-Net V3 ya no sigue una estructura simétrica como tal, hay diferencias en la cantidad de filtros que usa cada capa, en la primera etapa del codificador se tienen 2 capas, la primera formada por una CP de 250x250x9 y la segunda capa conformada por una CPS de 250x250x4 seguido por un *up-sampling* de 2x2, la

segunda etapa cuenta con 2 capas más, comenzando por una CP de 500x500x4 seguida por una CPS de 500x500x2. El proceso de decodificación busca expandir gradualmente las dimensiones mientras se recupera la información esencial perdida durante la etapa de codificación.

Salida: es una capa con una CT que usa función de activación *ReLU* definida por la ecuación (1.2) para que las características definidas por la FCN tomen valores positivos. Mediante la función *ReLU*, se genera una salida no lineal que activa las características definidas por la FCN, asegurando que tomen valores positivos. Como en versiones anteriores, el siguiente paso es segmentar la imagen, pero en el caso de M-Net V3, se utilizó un umbral fijo de 0.5 para imágenes normalizadas. El umbral fijo fue definido debido al propio *TensorFlow*, el fondo toma valores cercanos a cero mientras que el microorganismo de interés toma valores cercanos a 1, *Tensorflow* trunca valores iguales o menores a 0.5 y caso contrario los redondea. De esta manera se generan *blobs* que representan los microorganismos de interés.

3.3.4 Muestras de resultados del modelo

La Figura 3.13 muestra los resultados cualitativos obtenidos por el modelo y de acuerdo con ellos, se interpreta lo siguiente:

CA. La morfología resultante es adecuada, si bien ahora tiene el detalle de que se notan muchos artefactos en el resultado final, es muy sencillo separar los elementos de interés por medio de una binarización por medio de umbral fijo.

EC. La morfología es adecuada, al igual que en CA, la morfología corresponde al elemento identificado, el modelo no se limita a poner puntos como en versiones anteriores, si bien tiene el mismo inconveniente de los artefactos también se eliminan como en CA.

SA. La morfología obtenida es adecuada, a simple vista parece que el modelo no detecta de manera correcta, pero eso es un problema de artefactos (como en CA y en EC) que se elimina al quitar el ruido.

El modelo M-Net V3 usa una pequeña cantidad de parámetros, lo cual lo vuelve atractivo y fácilmente ejecutable por dispositivos de hardware con recursos limitados. Además, este modelo prioriza la eficiencia. Dentro del contexto de esta investigación, se destaca como el más eficaz en la identificación de los elementos de interés presentes en la base de datos BYRI. La

clave para la creación de esta arquitectura liviana y eficaz radica en la capacidad del modelo para extraer y comprimir la información de manera selectiva, lo que asegura el procesamiento exclusivo de la información esencial.

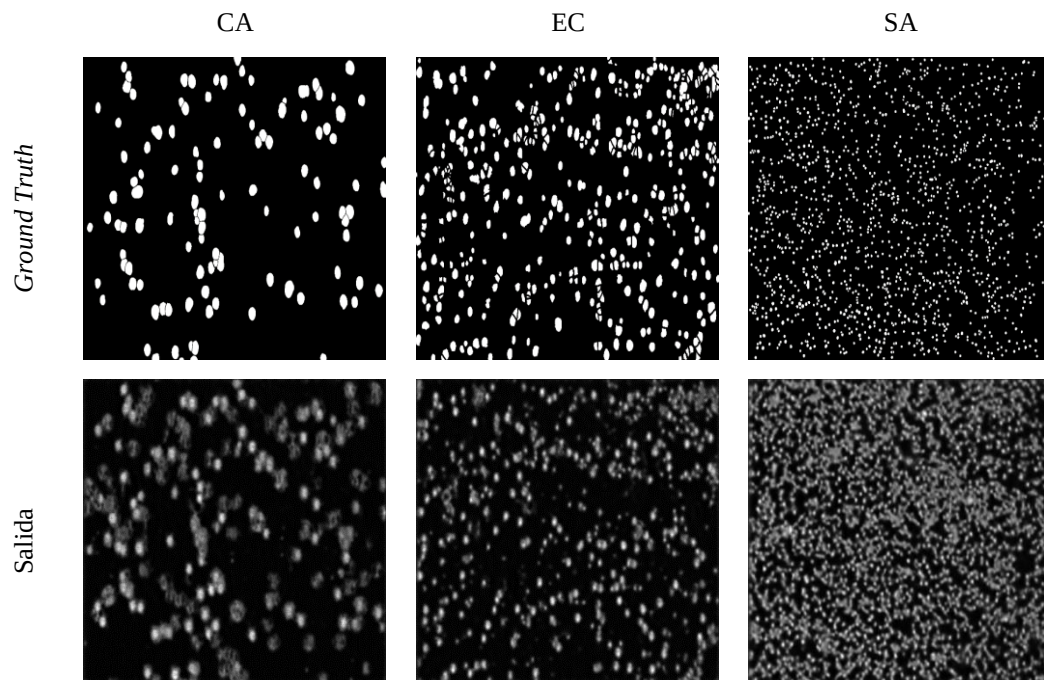


Figura 3.13 Resultados del modelo V3.

3.4 Función de pérdida

Una función de pérdida (*loss function*), es una función que evalúa la diferencia de las predicciones realizadas por una red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Así, es posible cuantificar la discrepancia entre las predicciones de la red y los resultados reales, y para reducirla, se procede a ajustar los diversos pesos de la red neuronal [35].

Para el entrenamiento del modelo M-Net (en las 3 versiones) se utilizó *Binary Focal Cross Entropy* [36], debido a que tiene un parámetro llamado de “enfoque” (γ), que permite que las muestras que son difíciles de clasificar se penalicen más en comparación con los ejemplos fáciles. Para ello, *Binary Focal Cross Entropy* introduce dos términos adicionales en comparación con la *Binary Cross Entropy* convencional [36]. La ecuación que define la función de pérdida es la siguiente:

$$loss(y, \hat{y}) = -\alpha \cdot (1 - \hat{y})^\gamma \cdot y \cdot \log(\hat{y}) - (1 - \alpha) \cdot \hat{y}^\gamma \cdot (1 - y) \cdot \log(1 - \hat{y}) \quad (3.3)$$

Donde:

- $loss(y, \hat{y})$ es la pérdida resultante al comparar la etiqueta verdadera y con la predicción $\hat{y}$.
- α es el peso utilizado para ajustar el balance de clases. Puede ser un valor entre 0 y 1.
- γ es el parámetro de modulación focal, que controla el enfoque en muestras difíciles. Generalmente se utiliza un valor mayor a 0.
- y es la etiqueta verdadera, que puede ser 0 o 1.
- $\hat{y}$ es la predicción del modelo, que representa la probabilidad estimada de que la muestra pertenezca a la clase positiva (1).

El primer término pondera las predicciones incorrectas de la clase positiva por un factor $(1 - \hat{y})^\gamma$, mientras que el segundo término pondera las predicciones incorrectas de la clase negativa por un factor $\hat{y}^\gamma$. Estos términos permiten que la función de pérdida se centre más en las muestras difíciles, ayudando a mitigar el impacto del desequilibrio de clases y mejorando el rendimiento en casos de clasificación desafiante. La *Binary Focal Cross Entropy* es una herramienta útil en problemas de clasificación binaria donde se requiere manejar el desequilibrio de clases como es el caso de las muestras de BYRI, donde el conjunto de los píxeles de fondo es mucho mayor que el conjunto de píxeles que representan los microorganismos [35].

4 IMPLEMENTACIÓN EN JETSON

M-Net se diseñó con bajo costo computacional para utilizarlo en cualquier servidor, computadora o sistemas embebido apoyados con Unidades Graficas de Procesamiento (GPU). Por ello, en este capítulo se presenta un análisis de la implementación de M-Net en un sistema GPU embebido.

Este capítulo se organiza de la siguiente manera. La sección **4.1** reporta los conceptos de sistemas embebidos, la sección **4.2** describe la arquitectura de la GPU embebida Jetson TX2 y la sección **4.3** presenta la implementación de M-Net en la Jetson TX2.

4.1 Conceptos de sistemas embebidos

En los sistemas embebidos actuales, existen tres arquitecturas: las Unidades de Procesamiento Gráfico (GPU) embebidas, las arquitecturas de procesadores conocidas como *Complex Instruction Set Computer (CISC)* y *Reduced Instruction Set Computer (RISC)*.

4.1.1 GPU embebida

Una GPU embebida es un sistema embebido que incorpora una o más GPUs en su diseño. Las GPUs son componentes especializados diseñados para acelerar el procesamiento de gráficos, como renderización 3D, simulaciones físicas y procesamiento de imágenes. Aunado a esto, conforme las GPUs se ha vuelto más potentes y versátiles, también comenzaron a utilizarse para acelerar el aprendizaje de máquina y la ciencia de datos [37]. Un sistema GPU embebido suele ser más pequeño y más integrado que las GPUs tradicionales que se encuentran en las computadoras de escritorio. Estos sistemas se utilizan en dispositivos donde la eficiencia energética, el tamaño compacto y la capacidad de procesamiento son críticos [37].

4.1.2 CISC

Los sistemas tipo CISC son arquitecturas de procesadores que se caracterizan por tener un conjunto de instrucciones amplio y complejo. Estas arquitecturas se diseñaron originalmente para ejecutar instrucciones muy variadas y detalladas en un solo ciclo de reloj, lo que permitía a los programadores escribir códigos más concisos y realizar tareas más complejas con menos instrucciones [38] [39]. Las características distintivas de los sistemas tipo CISC incluyen:

Conjunto de amplio de instrucciones: las arquitecturas CISC ofrecen una amplia gama de instrucciones para realizar operaciones muy específicas y detalladas.

Modos de direccionamiento múltiple: las arquitecturas CISC admiten varios modos de direccionamiento para acceder a los datos en la memoria de diferentes formas. Esto puede incluir modos directos, indirectos, indexados y etc.

Instrucciones de longitud variable: las instrucciones en las arquitecturas CISC pueden tener diferentes longitudes en bytes debido a la variedad de operaciones que pueden realizar.

Ejecución microprogramada: los procesadores CISC utilizan un nivel de microcódigo para ejecutar las instrucciones complejas. Esto significa que las instrucciones del lenguaje de programación se traducen a una secuencia de instrucciones internas que el procesador ejecuta.

Uso intensivo de la memoria: debido a la complejidad de las instrucciones, los programas escritos en lenguaje ensamblador CISC pueden requerir más espacio de memoria que sus contrapartes en arquitecturas RISC.

Ejemplos de arquitecturas CISC incluyen las familias de procesadores Intel x86 (como el Intel Core) y la serie de procesadores AMD.

4.1.3 RISC

Los sistemas tipo RISC son un tipo de arquitectura de procesadores que se caracteriza por tener un conjunto de instrucciones reducido y optimizado para ejecutar instrucciones simples en un número fijo de ciclos de reloj. A diferencia de las arquitecturas CISC, RISC se centran en la simplicidad y en la ejecución eficiente de instrucciones comunes [38] [39]. Las características clave de las arquitecturas RISC son:

Conjunto de instrucciones reducido: Las arquitecturas RISC tienen un conjunto de instrucciones más pequeño y uniforme en comparación con las CISC. Esto facilita la implementación del hardware y simplifica la ejecución de instrucciones.

Instrucciones simples: Las instrucciones en las arquitecturas RISC son más simples y realizan operaciones básicas en un solo ciclo de reloj. Cada instrucción se diseña para ejecutarse en un tiempo predecible y constante.

Ejecución en varios ciclos: Aunque las instrucciones RISC son más simples, a menudo se requieren varios ciclos de reloj para ejecutar una instrucción. Sin embargo, este enfoque permite que las instrucciones se ejecuten de manera más predecible y eficiente.

Uso de registros: Las arquitecturas RISC utilizan registros internos para almacenar datos temporales, minimizando la necesidad de acceder constantemente a la memoria principal. Esto acelera las operaciones y mejora el rendimiento.

Pipeline eficiente: Las arquitecturas RISC son propicias para la implementación de pipelines, que dividen la ejecución de instrucciones en varias etapas para lograr un mayor rendimiento y paralelismo.

Optimización de compilador: Las instrucciones RISC están diseñadas para ser fáciles de traducir desde lenguajes de alto nivel a lenguaje ensamblador. Esto permite que los compiladores generen código eficiente y predecible.

Ejemplos de arquitecturas RISC incluyen el *Microprocessor without Interlocked Pipeline Stages* (MIPS), *Advanced RISC Machine* (ARM) y *Scalable Processor Architecture* (SPARC). Aunque históricamente las diferencias entre CISC y RISC eran más marcadas, las tecnologías modernas han permitido la convergencia entre ambas arquitecturas. Incluso, muchas implementaciones actuales incorporan elementos CISC y RISC para lograr un equilibrio entre complejidad y eficiencia [38] [39]. La Tabla 4.1 muestra las diferencias cruciales que presentan estas arquitecturas.

Tabla 4.1 Resumen de arquitecturas CISC y RISC.

	CISC	RISC
Énfasis	<i>Hardware</i>	<i>Software</i>
Instrucciones	Instrucciones complejas multi-reloj	Un solo reloj, instrucciones reducidas
Memoria	Memoria a memoria: "CARGAR" y "ALMACENAR", incorporado en las instrucciones	Registro a Registro: "CARGAR" y "ALMACENAR", son instrucciones independientes
Tamaño del código y ciclos	Tamaños de código pequeños, muchos ciclos por segundo	Pocos ciclos por segundo, tamaños de código grandes

4.2 Jetson TX2

La Jetson TX2 es una plataforma de cómputo embebida de NVIDIA diseñada para aplicaciones de inteligencia artificial y robótica. A continuación, se presentan sus principales especificaciones y arquitectura:

- GPU: NVIDIA Pascal con 256 núcleos CUDA
- CPU: ARM Cortex-A57 de 64 bits de cuatro núcleos
- RAM: 8 GB de memoria LPDDR4
- Almacenamiento: 32 GB de almacenamiento eMMC
- Conectividad: Wi-Fi 802.11ac, Bluetooth
- Interfaces: USB 3.0, HDMI, Ethernet, PCIe
- Soporte para cámaras y sensores: MIPI CSI-2 (interfaces de cámara), GPIO, I2C, I2S, UART, SPI.

La Jetson TX2 se basa en la arquitectura NVIDIA Pascal [40] conocida por su eficiencia energética y capacidades de procesamiento. La combinación de núcleos CUDA y CPU ARM permite un rendimiento sólido para cálculos de IA y en tareas generales. La Tabla 4.2 presenta las diferencias entre PC y Jetson TX2.

Tabla 4.2 Especificaciones de PC utilizada para el entrenamiento.

Especificaciones de la PC	Detalles
CPU	Intel Xeon CPU E5620
RAM	9 GB DDR3
GPU	NVIDIA GeForce RTX 3060

- **GPU:** Tiene 256 núcleos CUDA, mientras que la GPU RTX 3060 para PC tiene 3584 núcleos CUDA. La RTX 3060 está diseñada principalmente para juegos y aplicaciones de cómputo de propósito general, lo que la hace más poderosa en términos de capacidad de procesamiento de gráficos.
- **CPU:** La CPU de la Jetson TX2 es un ARM Cortex-A57 de cuatro núcleos, mientras que la PC cuenta con un procesador Intel Xeon E5620. Los procesadores Xeon suelen ser más potentes en tareas de procesamiento intensivo y manejo de múltiples hilos,

lo que podría darle a la PC una ventaja en tareas generales y de cálculo de propósito general.

- **RAM:** La Jetson TX2 tiene 8 GB de RAM LPDDR4, mientras que la PC tiene 9 GB de RAM DDR3. En términos de velocidad y eficiencia, la RAM LPDDR4 tiende a ser más rápida y consumir menos energía que la DDR3.

Las CPUs, especialmente aquellas diseñadas para servidores y dispositivos móviles, tienen una arquitectura de propósito general y están optimizadas para una amplia gama de tareas, incluyendo cálculos de punto flotante y enteros, si desea ver o analizar más acerca de los núcleos, procesamiento, TFLOPS, etc de una diversa cantidad de Procesadores, se le recomienda visitar [41]. Por otro lado, las GPU están diseñadas para renderizar gráficos y procesar grandes cantidades de datos en paralelo (gráficos de videojuegos y aplicaciones científicas) y poseen un gran número de núcleos (cientos o miles) que están optimizados para realizar cálculos en paralelo [41].

4.3 Implementación de M-Net V3 en la Jetson TX2

El modelo fue generado en PC por medio de *TensorFlow*, con extensión *.h5* para almacenar la información creada del todo el modelo y sus respectivos [42]. Sin embargo, para extraer la información e implementarlo en un dispositivo embebido, se utilizó *Open Neural Network Exchange* (ONNX) [43], que genera un conjunto de operadores que modelan los bloques de construcción de modelos de aprendizaje máquina y profundo con un formato compatible para diferentes plataformas de hardware. ONNX lee el archivo de *Tensorflow* con extensión *.h5* de Python y lo convierte a un archivo con extensión *.onnx*.

4.3.1 Funcionamiento en tarjeta de desarrollo Nvidia Jetson TX2

ONNX puede funcionar de 2 modos: CPU o GPU: esta dualidad de enfoques permite que los desarrolladores optimicen la ejecución de modelos según las demandas específicas de la aplicación. El modelo M-Net V3 en formato *.onnx* tiene 3 procesos para generar el resultado final como se muestra en la Figura 4.1. Dicho proceso tiene la siguiente secuencia.

Inicio → Figura 4.1 a) → Figura 4.1 b) → Figura 4.1 c) → Fin

Inicio: Inicio del programa y se enfoca a la adquisición de la imagen de entrada.

Figura 4.1 a). Salida que tiene el modelo M-Net con la imagen de entrada.

Figura 4.1 b). Segmentar para obtener los *blobs* que representan los microorganismos de interés (ya sea por medio de OTSU o Umbral fijo).

Figura 4.1 c). Se realiza un conteo de los *blobs* que se obtuvieron. Luego, se utiliza el perímetro de los blobs como coordenadas y se imprimen en la imagen original. Por último, se concatena de nuevo la imagen original del lado izquierdo.

Fin: Fin del procesamiento para el conteo de microorganismos.

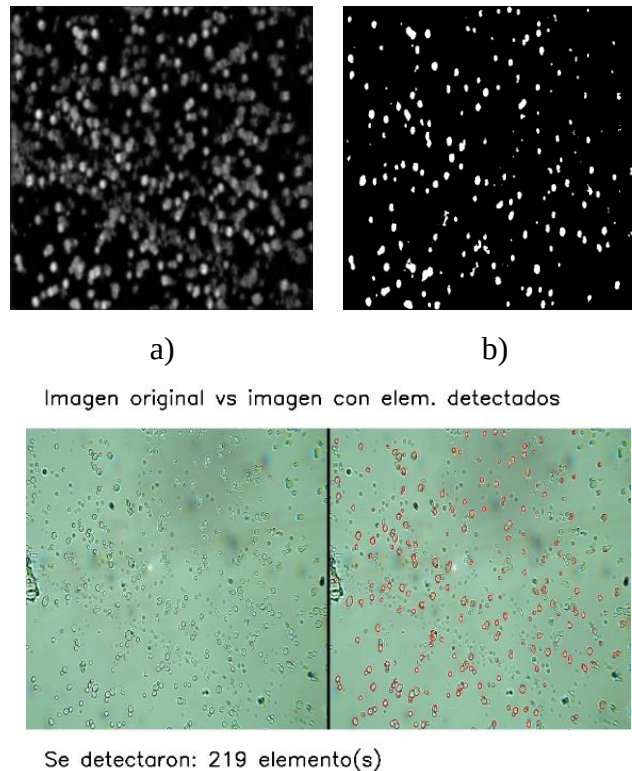


Figura 4.1 Resultados con M-Net V3

a) Salida del modelo b) Salida del modelo binarizada c) Comparación de la entrada con los resultados encontrados (encerrados en rojo).

4.3.2 CPU

En esta sección se analizará M-Net V3 y su rendimiento utilizando la CPU y GPU de un sistema de cómputo discreto. En la Figura 4.2 se muestra los resultados de los 3 procesos que se abordan en la Figura 4.1.

```

100%|██████████| 1502/1502 [02:27<00:00, 10.20it/s]
100%|██████████| 1502/1502 [00:16<00:00, 90.22it/s]
100%|██████████| 1502/1502 [02:27<00:00, 10.21it/s]

```

Figura 4.2 Tiempo de generación de resultados usando CPU.

El tiempo de procesamiento y el consumo energético son dos aspectos de suma relevancia, ya que se busca lograr dispositivos que combinen eficiencia en el procesamiento de cómputo con un consumo energético optimizado. Para analizar el consumo energético en la tarjeta de desarrollo NVIDIA Jetson TX2 se utilizó el *software* llamado *jetson_stats* o *jtop* que permite monitorear en tiempo real el uso de recursos que el dispositivo tiene (RAM, CPU, GPU, Consumo energético, entre otros). De acuerdo con dicho *jtop*, el consumo que tiene el dispositivo mientras corre el modelo de M-Net V3 en formato *.onnx* es de 5 W en promedio mientras se realiza la predicción (Figura 4.3), lo que quiere decir que el consumo energético usando la CPU es ínfimo considerando que el rango energético de la tarjeta de desarrollo NVIDIA Jetson TX2 ronda entre los 5-15 W.

[Power]		[Inst]	[Avg]
1V8	IO	0mW	0mW
3V3	IO SLP	0mW	0mW
3V3	SYS	40mW	39mW
3V3	SYS M2	0mW	0mW
4V0	WIFI	0mW	0mW
5V0	IO SYS	2.1W	2.1W
	MUX	2.5W	2.5W
	SYS CPU	2.3W	2.3W
	SYS DDR	1.1W	1.1W
	SYS GPU	153mW	157mW
	SYS SOC	690mW	689mW
	VDD_IN	5.0W	5.0W

Figura 4.3 Tabla de consumo energético de tarjeta de desarrollo Nvidia Jetson TX2 mientras corre el modelo M-Net V3 obtenido con *jtop* (CPU).

4.3.3 GPU

El uso de GPU no tiene impacto significativo al ejecutar el programa usando la extensión *.onnx*, ya que tiempo de procesamiento por imagen y el consumo de potencia son afectados en

grado mínimo, registrando valores prácticamente invariables, tal como se refleja en la Figura 4.4. Esto puede ser debido a que el modelo M-Net V3 tiene una cantidad pequeña de parámetros que no genera un cambio significativo para el *hardware* si se trabaja de forma secuencial o paralela.

```
100%|██████████| 1502/1502 [02:30<00:00, 9.99it/s]
100%|██████████| 1502/1502 [00:17<00:00, 83.74it/s]
100%|██████████| 1502/1502 [02:31<00:00, 9.92it/s]
```

Figura 4.4 Tiempo de generación de resultados usando GPU.

El consumo energético se obtiene con el *software* *jtop* y de acuerdo con la Figura 4.5, es de 5.1 W (recuadro rojo), lo que quiere decir que a pesar de usar la GPU y contar con un procesamiento en paralelo no se redujo los tiempos, al contrario, el tiempo final de procesamiento aumento 8 seg. Si bien el modelo M-Net V3 tiene matrices en donde el procesamiento por medio de GPU es más eficiente (convoluciones profundas y convoluciones tradicionales), el modelo cuenta con multiplicaciones matriciales y vectoriales (convoluciones profundas separables) en donde en los vectores la GPU por cuestiones de arquitectura se obliga a convertirlas a matriz para procesarlas.

Los resultados obtenidos pueden estar influidos por una combinación de factores tales como el tamaño del modelo, la arquitectura del modelo, la eficiencia en la transferencia de datos, la implementación del software y las capacidades particulares de la tarjeta NVIDIA Jetson TX2. La GPU tiene un consumo mayor que la CPU y los tiempos también resultan elevados en comparación con la CPU, si bien anteriormente se explicó causas a tener en cuenta, se menciona en [44], [45] y [46] posibles inconvenientes del hardware por los que el aumento de tiempo no se deba a una causa en específico, puede que el problema radique en un conjunto de inconvenientes que limitan el proceso y la transferencia de datos, por ejemplo la velocidad de acceso de la memoria RAM y ROM, eficiencia de drivers, etc., a eso sumado lo anterior, el diseño del modelo, software utilizado, etc. La Tabla 4.3, muestra un resumen de los resultados obtenidos usando CPU y GPU además de una breve explicación de algunos problemas puntuales que se pudo generar al hacer uso de cada arquitectura.

Tabla 4.3 Comparación de resultados entre CPU y GPU [44] [45] [46]

Aspecto	CPU	GPU
Tiempo de Ejecución	310 segundos	318 segundos
Consumo Promedio	5.0 W	5.1 W
Imágenes por Segundo	4.8 img/s	4.7 img/s
Paralelismo y Recursos	-	Posiblemente limitado por la arquitectura más antigua y recursos limitados en la tarjeta NVIDIA Jetson TX2
Optimizaciones Específicas	-	Posiblemente optimizaciones específicas en la implementación de software o controladores
Transferencia de Datos	Posiblemente más eficiente en la CPU debido a la arquitectura de la memoria	Posiblemente acceso a memoria más tardado o de manera indirecta por medio de la CPU
Implementación del Modelo	Posiblemente más optimizada para la CPU debido al uso de convoluciones profundas separables.	Procesamiento más eficiente en convoluciones profundas y convoluciones tradicionales.

[Power]	[Inst]	[Avg]
1V8 IO	0mW	0mW
3V3 IO SLP	0mW	0mW
3V3 SYS	40mW	36mW
3V3 SYS M2	0mW	0mW
4V0 WIFI	19mW	14mW
5V0 IO SYS	2.1W	2.1W
MUX	2.5W	2.5W
SYS CPU	2.4W	2.4W
SYS DDR	1.1W	1.1W
SYS GPU	153mW	153mW
SYS SOC	690mW	688mW
VDD_IN	5.1W	5.1W

Figura 4.5 Tabla de consumo energético de tarjeta de desarrollo Nvidia Jetson TX2 mientras corre el modelo M-Net V3 obtenido con jtop (GPU).

4.3.4 Diferencia entre usar *Tensorflow* y *Onnx*

Un aspecto interesante es comparar las diferencias entre M-Net V3 con extensión *.h5* de *TensorFlow* y *.onnx*. Si bien existen diferencias [43] con respecto a los resultados que estos formatos generan, de acuerdo con la gráfica en la Figura 4.6, el 90% de las muestras tienen el mismo resultado con ambas extensiones. El 10% de muestras que son diferentes tienen diferencias de ± 2 microorganismos, como se ve en la Tabla 4.4. La clase más que puede resultar más afectada es *Candida albicans* debido a que es la clase en la base de datos BYRI que tiene menos microorganismos por imagen, debido a esto puede que llegue a ser necesario fijar un mínimo de microorganismos mínimo por imagen o realizar promediados para validar los resultados.

Con respecto a la Tabla 4.4, la información que se tiene de cada imagen esta codificada de la siguiente manera:

- De la imagen **img_000001.png** a la imagen **img_000200.png** pertenecen a la clase *Candida albicans*.
- De la imagen **img_000201.png** a la imagen **img_000400.png** pertenecen a la clase *Escherichia coli*.
- De la imagen **img_000401.png** a la imagen **img_000600.png** pertenecen a la clase *Staphylococcus aureus*.

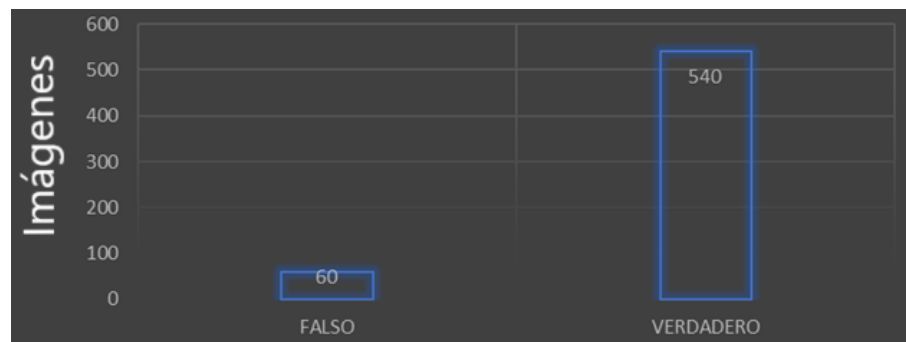


Figura 4.6 Comparación de resultados con 2 formatos distintos (*Tensorflow* y *onnx*), los casos registrados como verdaderos son resultados iguales entre formatos, los resultados evaluados como falsos son resultados distintos.

Tabla 4.4 Resultados en donde el modelo de Tensorflow y Onnx generan resultados distintos (algunos ejemplos.).

Nombre de imagen	<i>Tensorflow</i>	ONNX
img_000054.png	52	51
img_000086.png	151	150
img_000088.png	153	154
img_000089.png	155	154
img_000115.png	149	150
img_000219.png	158	159
img_000247.png	543	544
img_000248.png	543	544
img_000250.png	478	479
img_000251.png	478	479
img_000574.png	922	921
img_000575.png	989	988
img_000581.png	907	905
img_000583.png	916	915
img_000584.png	1011	1010

5 RESULTADOS

Este presenta los experimentos y resultados que generan los modelos M-Net. De esta manera, la sección 4.1 reporta los resultados cualitativos de M-Net. La sección 4.2 presenta los resultados cuantitativos de las tres versiones de M-Net. Finalmente, la sección 4.3 describe un análisis de significancia estadística de los resultados.

5.1 Proceso de generación de conteo

Como se mencionó en el capítulo anterior, se tiene un posprocesamiento con la imagen obtenida del modelo M-Net V3 el cual lleva un procedimiento para generar el conteo de microorganismos.

5.1.1 Procesamiento de los datos

En las Figura 5.1, Figura 5.2 y Figura 5.3 se muestra la salida que se genera de los modelos M-Net en sus 3 versiones, estas imágenes mostradas son imágenes a escala de grises.

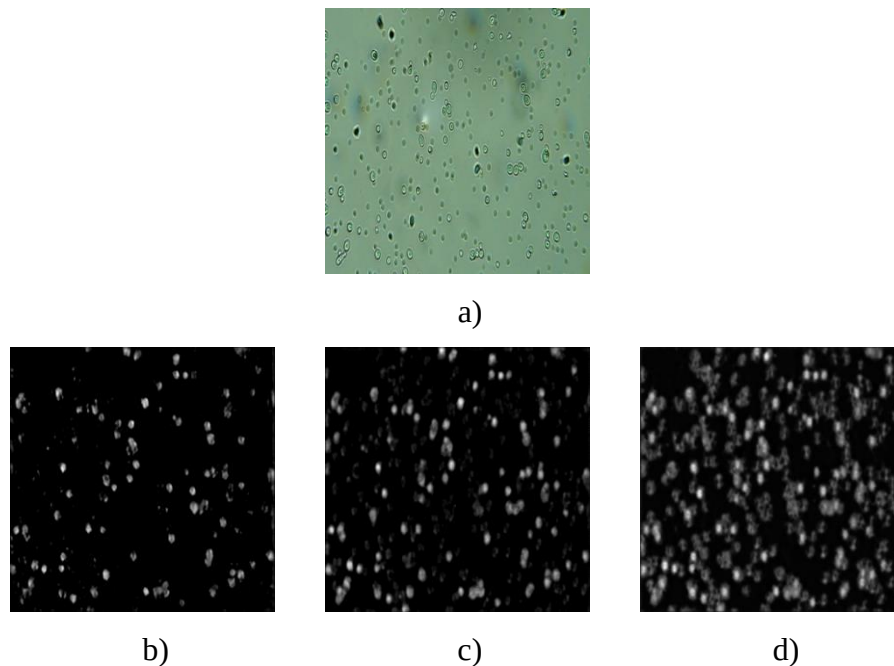


Figura 5.1 Resultados obtenidos en el procedimiento de Predicción (Clase Candida Albicans), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), Resultado de la imagen de ejemplo (M-Net V3).

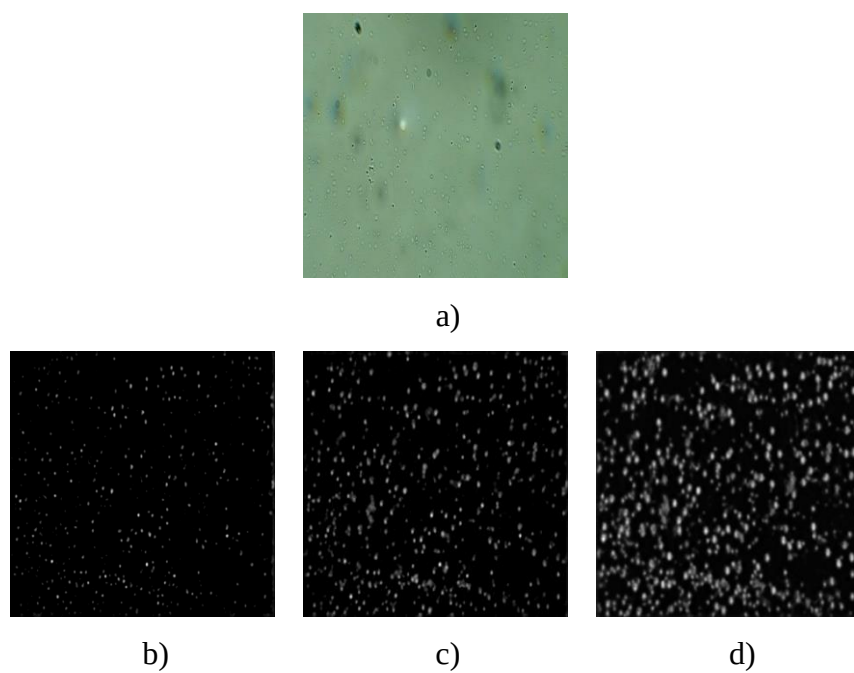


Figura 5.2 Resultados obtenidos en el procedimiento de Predicción (Clase Escherichia coli), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3)

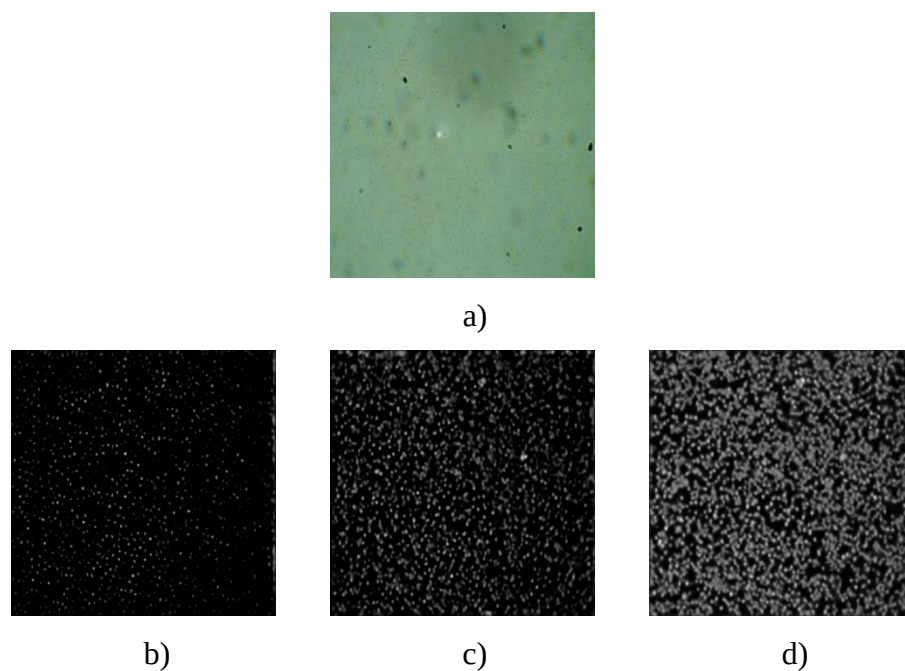


Figura 5.3 Resultados obtenidos en el procedimiento de Predicción (Clase Staphylococcus aureus), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3)

5.1.2 Umbralización

En las Figura 5.4, Figura 5.5 y Figura 5.6 se muestra el resultado al binarizar las imágenes de las Figura 5.1, Figura 5.2 y Figura 5.3, estas imágenes mostradas son imágenes binarias.

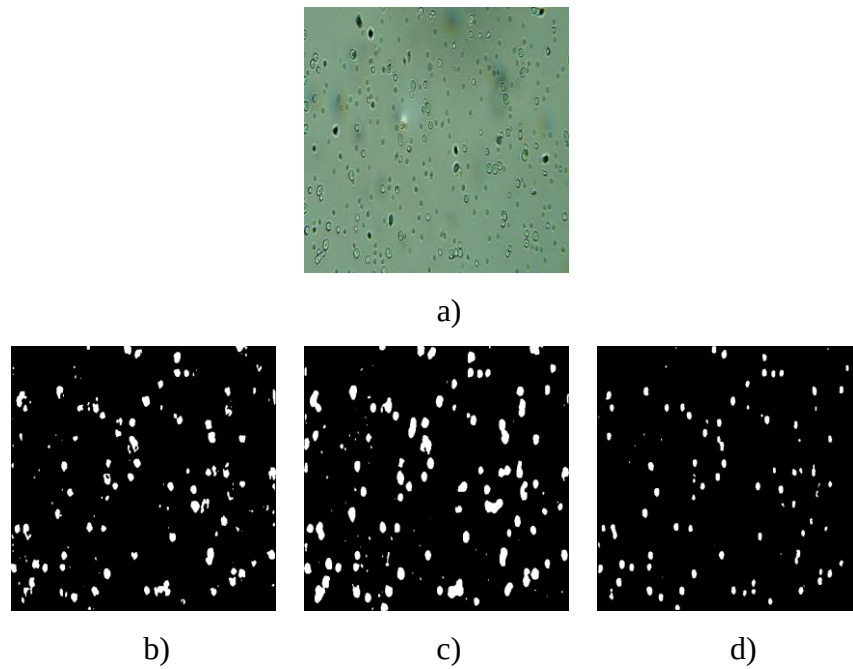


Figura 5.4 Resultados obtenidos en el procedimiento de Umbralización (Clase Candida Albicans), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3)

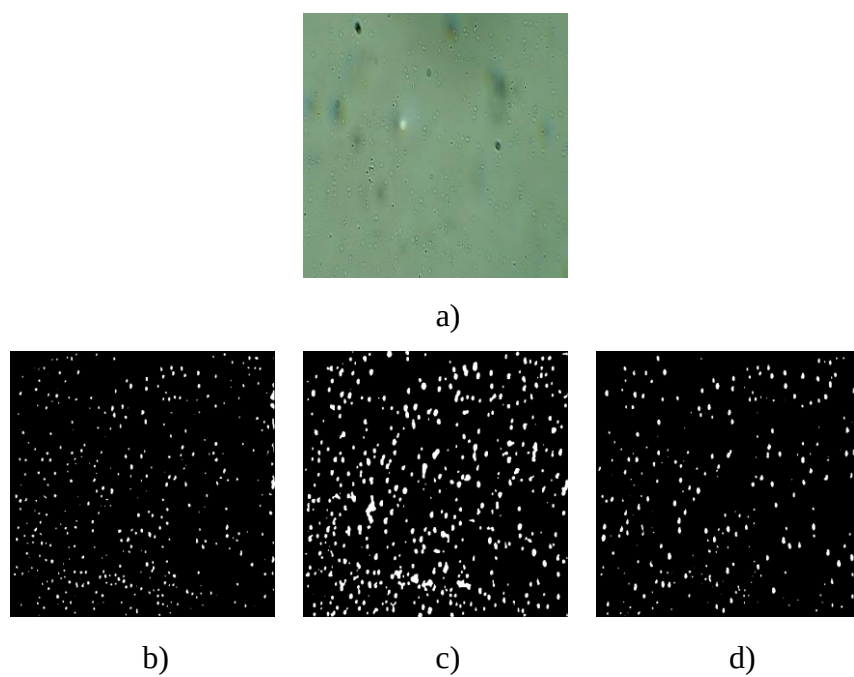


Figura 5.5 Resultados obtenidos en el procedimiento de Umbralización (Clase *Escherichia coli*), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3)

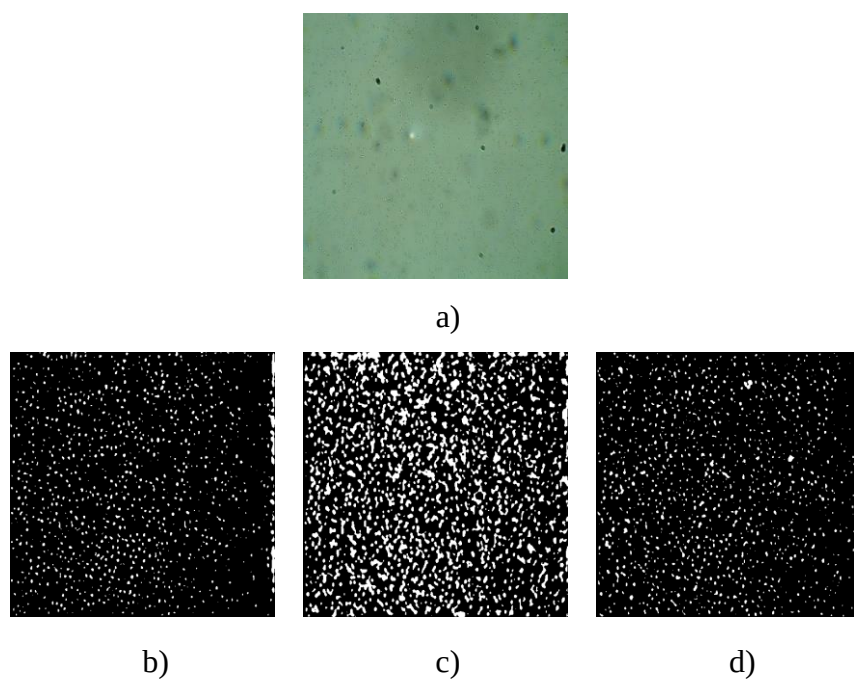


Figura 5.6 Resultados obtenidos en el procedimiento de Umbralización (Clase *Staphylococcus aureus*), a) Imagen de ejemplo, b) Resultado de la imagen de ejemplo (M-Net V1), c) Resultado de la imagen de ejemplo (M-Net V2), d) Resultado de la imagen de ejemplo (M-Net V3)

5.1.3 Análisis cualitativo

Los modelos M-Net se diseñaron con el objetivo de contar microorganismos para aplicaciones de análisis y cuantificación de microbiología (como lo son la citometría de flujo y la reacción de cadena de la polimerasa (PCR)). Este instrumento consiste en tomar los blobs de la salida de M-Net para contar los microorganismos presentes en la muestra de cultivo y generar un perímetro que se superpone a $I(x, y)^{RGB}$ para generar las imágenes $E(x, y)^{RGB}$. $E(x, y)^{RGB}$ se concatena con $I(x, y)^{RGB}$ para formar el $O(x, y)^{RGB} = \{I(x, y)^{RGB}, E(x, y)^{RGB}\}$ formando resultados como los mostrados en la Figura 5.7, la Figura 5.8 y la Figura 5.9, que resalta los microorganismos con un perímetro rojo. La cantidad de microorganismos encontrados por cualquier FCN en una imagen se le conoce como $c_{FCN, l}$, donde $FCN = \{MNet V1, MNet V2, MNet V3, ResNet50, MobileNet, Unet\}$, l se refiere al conjunto de imágenes. La cantidad de microorganismos encontrados en el GT se le conoce como $c_{FCN, l}$.

De acuerdo con los GTs, existen falsos positivos y falsos negativos en las 3 versiones tal como se ve en las Tabla 5.1, Tabla 5.2 y Tabla 5.3, pero el modelo M-Net en su V3 es el que tiene el índice más bajo con respecto a los falsos positivos y negativos, esto se puede apreciar al ver las Figura 5.7, la Figura 5.8 y la Figura 5.9 junto con las Tabla 5.1, Tabla 5.2 y Tabla 5.3, en donde el modelo M-Net V3 a diferencia de sus predecesores detecta los microorganismos con un mínimo de falsos positivos.

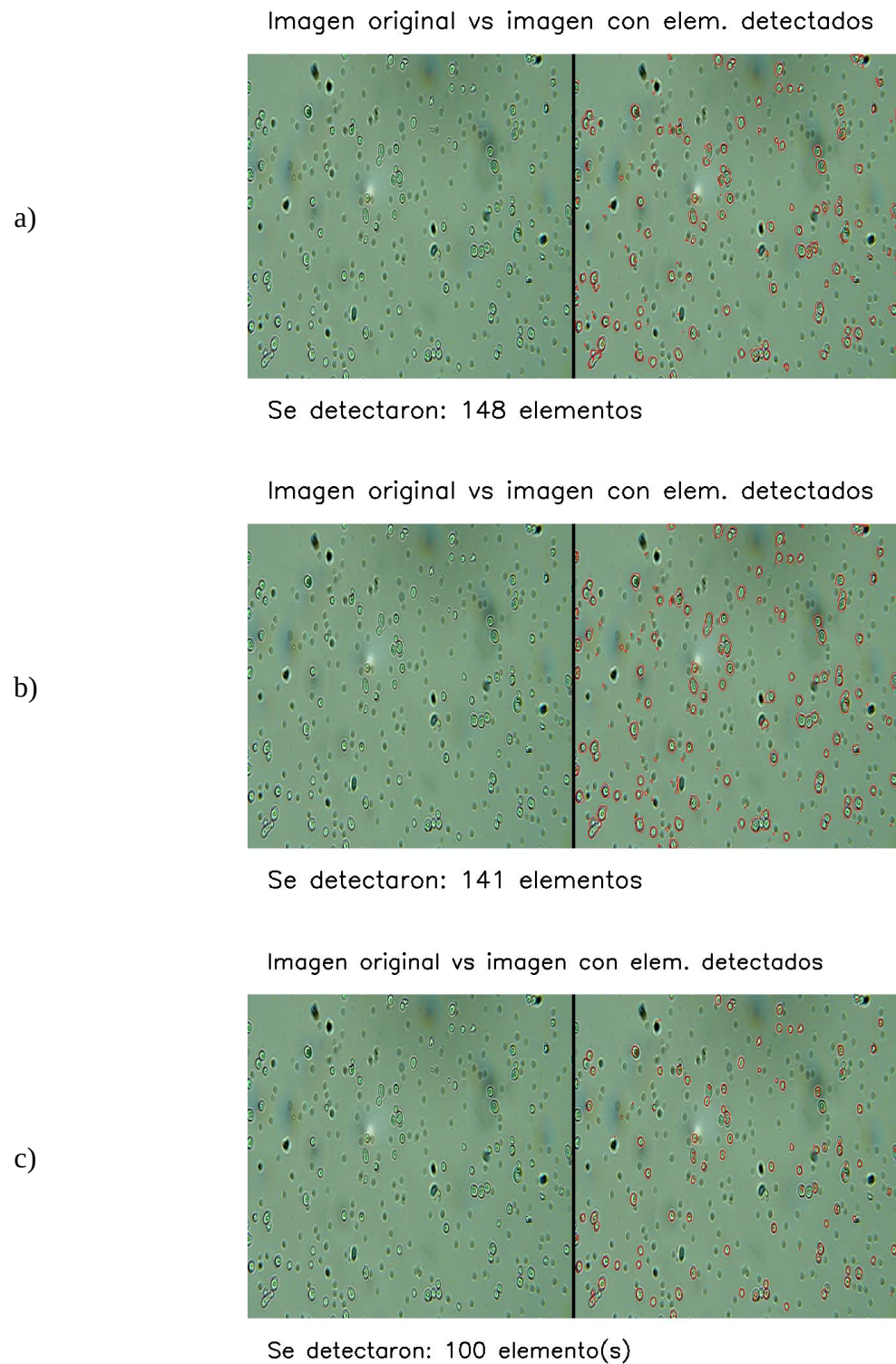
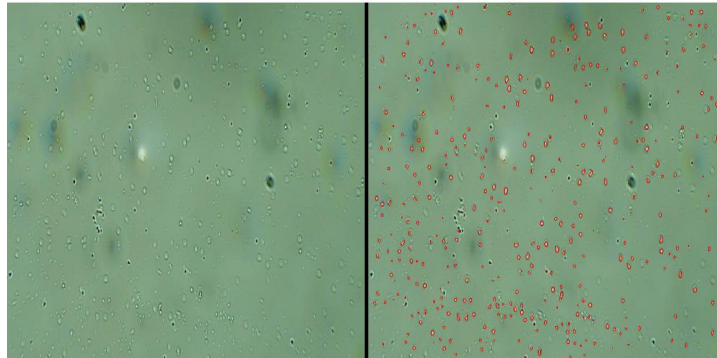


Figura 5.7 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase candida albicans).

Imagen original vs imagen con elem. detectados

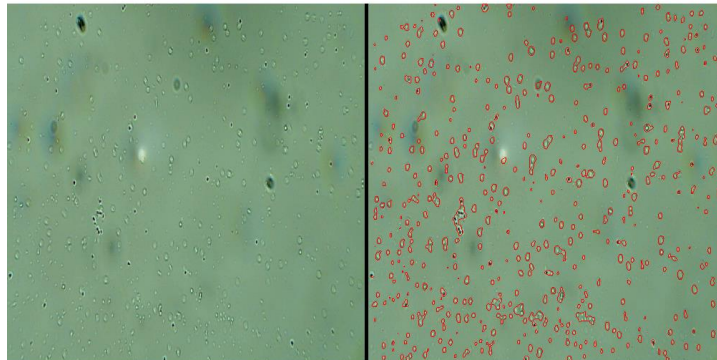
a)



Se detectaron: 393 elementos

Imagen original vs imagen con elem. detectados

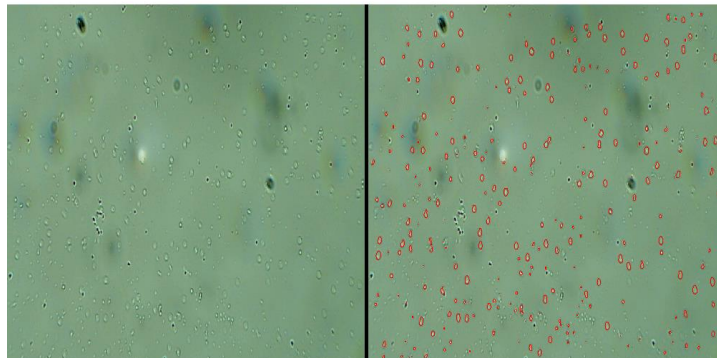
b)



Se detectaron: 447 elementos

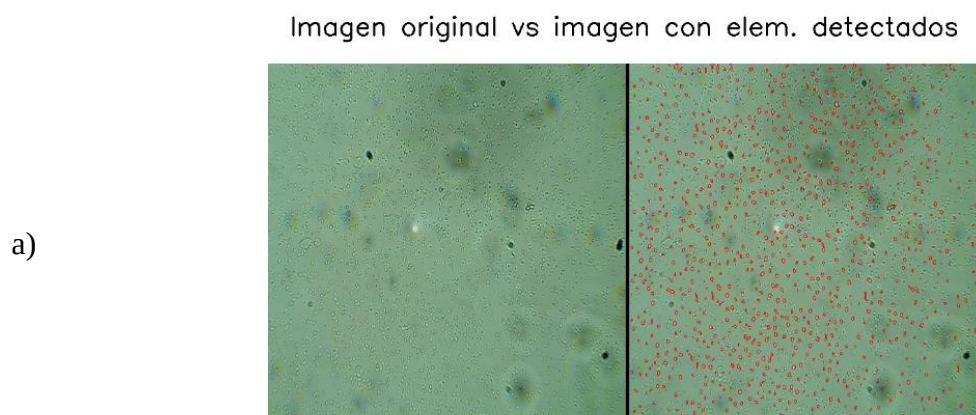
Imagen original vs imagen con elem. detectados

c)

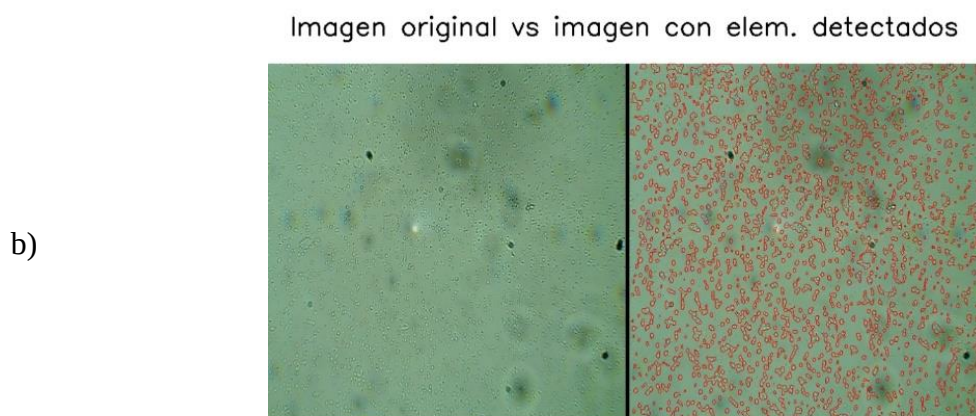


Se detectaron: 275 elemento(s)

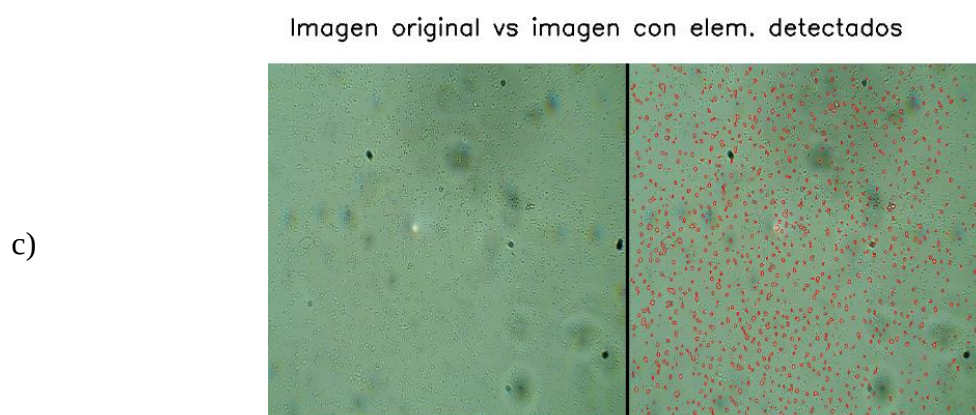
Figura 5.8 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase escherichia coli).



Se detectaron: 1032 elementos



Se detectaron: 1023 elementos



Se detectaron: 1119 elemento(s)

Figura 5.9 Elementos encontrados a) M-Net V1, b) M-Net V2, c) M-Net V3 (clase *staphylococcus aureus*).

Tabla 5.1 Resultados obtenidos de conteo empleando factor humano y modelos distintos (*Candida albicans*).

Candida Albicans							
Nombre de imagen	Persona (binarización)	v1 (Otsu)	v2 (Otsu)	v3 (binarización)	Mobilenet (binarización)	Resnet (binarización)	Unet (binarización)
0001.png	86	91	132	110	842	499	4427
0002.png	55	55	82	52	971	487	81
0003.png	57	53	77	45	996	487	79
0004.png	57	55	80	42	992	490	38
0005.png	52	51	70	47	996	496	60
0006.png	49	43	69	39	949	491	71
0007.png	47	44	73	37	982	490	83
0008.png	48	46	73	34	989	494	81
0009.png	44	45	74	31	974	484	107
0010.png	45	46	74	36	973	489	113

Tabla 5.2 Resultados obtenidos de conteo empleando factor humano y modelos distintos (*Escherichia coli*).

Escherichia Coli							
Nombre de imagen	Persona (binarización)	v1 (Otsu)	v2 (Otsu)	v3 (binarización)	Mobilenet (binarización)	Resnet (binarización)	Unet (binarización)
0201.png	188	145	353	137	952	476	3459
0202.png	568	315	644	508	1123	498	119
0203.png	672	315	644	508	1123	498	119
0204.png	207	144	361	147	964	479	3480
0205.png	570	304	672	456	1079	483	86
0206.png	720	304	672	456	1079	483	86
0207.png	199	140	336	154	948	484	3433
0208.png	685	281	662	433	1143	481	104
0209.png	658	281	662	433	1143	481	104
0210.png	207	141	349	149	970	477	3377

Tabla 5.3 Resultados obtenidos de conteo empleando factor humano y modelos distintos (*Staphylococcus aureus*).

Staphylococcus Aureus							
Nombre de imagen	Persona (binarización)	v1 (Otsu)	v2 (Otsu)	v3 (binarización)	Mobilenet (binarización)	Resnet (binarización)	Unet (binarización)
0401.png	397	286	647	398	979	490	183
0402.png	1205	355	882	955	1122	497	4866
0403.png	174	174	413	100	967	495	132
0404.png	407	269	642	389	976	489	199
0405.png	1152	345	868	1022	1139	503	3429
0406.png	181	186	461	222	960	490	170
0407.png	389	296	664	393	992	494	209
0408.png	1250	337	859	1079	1179	506	3448
0409.png	177	177	424	101	966	491	197
0410.png	384	287	637	405	975	489	208

5.2 Comparación de N-Net con otras FCN

La Tabla 5.4 y la Tabla 5.5 muestran una comparación las diferentes FCN que se analizaron en esta tesis.

En donde:

TP (True Positives): son los casos positivos que se predijeron correctamente.

TN (True Negatives): son los casos negativos que se predijeron correctamente.

FP (False Positives): son los casos negativos que se predijeron incorrectamente como positivos.

FN (False Negatives): son los casos positivos que se predijeron incorrectamente como negativos.

Las métricas seleccionadas fueron las siguientes:

Accuracy (Exactitud): La exactitud mide la proporción de predicciones correctas en el total de predicciones realizadas, la ecuación es la siguiente:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

Precision (Precisión): La precisión mide la proporción de predicciones positivas que fueron verdaderamente positivas, la ecuación es la siguiente:

$$Precision = \frac{TP}{TP + FP} \quad (5.2)$$

Recall (Recuperación o Sensibilidad): La recuperación mide la proporción de casos positivos verdaderos que se identificaron correctamente, la ecuación es la siguiente:

$$Recall = \frac{TP}{TP + FN} \quad (5.3)$$

F-score (Puntuación F): La puntuación F es una métrica que combina la precisión (ecuación (5.2)) y la recuperación (ecuación (5.3)) en una sola medida. Se calcula como la media armónica de precisión y recuperación, la ecuación es la siguiente:

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (5.4)$$

Intersection over Union (IoU): IoU se utiliza comúnmente en problemas de detección de objetos y segmentación de imágenes. Mide la superposición entre la predicción y la verdad fundamental (GT), la ecuación es la siguiente:

$$IoU = \frac{TP}{TP + FP + FN} \quad (5.5)$$

En este contexto, TP representa la cantidad de píxeles o áreas correctamente identificados en la predicción, FP son los píxeles o áreas que se identificaron incorrectamente y FN son los píxeles o áreas que se pasaron por alto.

Recuerda que estas ecuaciones son fundamentales en la evaluación del rendimiento de algoritmos de clasificación y detección en aprendizaje automático y visión por computadora. Asegúrate de tener los valores correctos de TP, TN, FP y FN para calcular estas métricas de manera precisa en tu problema específico.

De acuerdo con la Tabla 5.4 y la Tabla 5.5, el modelo M-Net V3 consigue un rendimiento que rivaliza con los resultados de las métricas y lo logra utilizando menos del 2% de parámetros en comparación con otros modelos. El que el modelo M-Net V3 tenga una cantidad baja de parámetros trae 2 fuertes beneficios, tiene un peso ligero en memoria de solo lectura (ROM) (el modelo pesa 12 Kilobytes en formato .onnx) lo cual lo hace fácil de implementar en dispositivos con recursos discretos y el uso de memoria RAM pico presenta menos de 800 Megabytes lo

cual nos indica el requerimiento máximo que va a tener el modelo, el modelo M-Net V3 es tan eficiente que al menos en cuestión de *hardware* es totalmente posible implementarlo en un microcontrolador.

Tabla 5.4 Comparación entre modelos M-Net, resultados finales del entrenamiento.

METRICAS COMPARATIVAS	M-Net v1	M-Net v2	M-Net v3
<i>Accuracy</i> (Exactitud)	0.953007	0.956434	0.955041
<i>F-score</i> (Puntuación F)	0.493019	0.549024	0.530829
<i>Precision</i> (Precisión)	0.854252	0.868926	0.851374
<i>Recall</i> (Recuperación o Sensibilidad)	0.354444	0.406175	0.392485
<i>Intersection over Union (IoU)</i>	0.643052	0.668975	0.660577
<i>Loss</i>	0.011496	0.009617	0.010261
Parámetros	66,025	45,484	575
Tiempo proc. (Hrs)	1.832139	0.670495	1.27241
RAM pico (GB)	4.14045	1.45176	0.769569
RAM prom (MB)	0.78833	0.650635	0.348877

Tabla 5.5 Comparación entre modelos del estado del arte, resultados finales del entrenamiento.

METRICAS COMPARATIVAS	ResNet-50 v2	MobileNet v2	Unet
<i>Accuracy</i> (Exactitud)	0.66116	0.431051	64.60185
<i>F-score</i> (Puntuación F)	0.440148	0.450228	0
<i>Precision</i> (Precisión)	0.632658	0.614109	0
<i>Recall</i> (Recuperación o Sensibilidad)	0.347418	0.364276	0
<i>Intersection over Union (IoU)</i>	0.615471	0.618748	0.46674
<i>Loss</i>	0.117107	0.120026	0.018268
Parámetros	Total params: 55,591,560, Trainable params: 32,026,760, Non-trainable params: 23,564,800	Total params: 22,284,744, Trainable params: 20,026,760, Non-trainable params: 2,257,984	34,513,475
Tiempo proc. (Hrs)	1.05201	1.2373	7.957502
RAM pico (GB)	1.365596	1.156067	4.364293
RAM prom (MB)	481.8398	238.5068	401.0239

5.3 Prueba de hipótesis

Una prueba de hipótesis es un método estadístico utilizado para determinar si una hipótesis sobre un parámetro poblacional está respaldada por los datos [47]. En esta tesis se consideró utilizar la prueba de hipótesis para conocer la variación estadística de los conteos del *blob* que obtienen las FCN respecto al conteo que se requiere en las técnicas de citometría de flujo o de PCR. De esta manera, se realizó una prueba de hipótesis de Lillifors [48] para conocer si los resultados de los conteos de las 1502 muestras forman o no una distribución normal para conocer si existieron sesgos durante la adquisición de los cultivos. Posteriormente, se realizó otra prueba de hipótesis de *Wilcoxon* [49] para determinar el grado de variación entre los resultados de las FCN y el conteo requerido.

5.3.1 Prueba de Lilliefors

La prueba de Lilliefors [48] tiene como hipótesis que la distribución de datos es normal. Lilliefors tiene un parámetro α que indica si dicha hipótesis se confirma o se rechaza. De esta manera, si $\alpha \geq 0.05$, la hipótesis se confirma, de manera que la distribución es normal. Por otro lado, si $\alpha < 0.05$, la hipótesis se rechaza [50]. Esta prueba se aplicó al conjunto de los 1502 conteos que se tiene en la base de datos para conocer si se debe hacer una prueba *t-Student* o *Wilcoxon* para conocer la significancia estadística.

Para realizar la prueba de Lilliefors, las 1502 imágenes se separaron por clase, generando tres variables aleatorias: conteos de EA, conteos de SA y conteos de CA. Dichas variables fueron probadas con las redes M-Net V1, M-Net V2, M-Net V3, ResNet-50 V2, MobileNet V2 y Unet. Se implementó la prueba de Lilliefors y en todos los casos, $\alpha = 0$, lo que quiere decir que la distribución de los datos no es normal. Esto se puede ver en las Figura 5.10, Figura 5.11, Figura 5.12, Figura 5.13, Figura 5.14, Figura 5.15, Figura 5.16, Figura 5.17, Figura 5.18, Figura 5.19, Figura 5.20, Figura 5.21, Figura 5.22, Figura 5.23, Figura 5.24, Figura 5.25, Figura 5.26 y Figura 5.27 donde el gráfico de normalidad muestra que dichas distribuciones distan bastante de ser normales [50].

Al analizar por qué las distribuciones fueron diferentes a una normal, una hipótesis para explicar este comportamiento es que contiene imágenes obtenidas por medio de video esto puede ser propenso a errores de muestreo, el video puede generar un sesgo en la selección de

muestras, las condiciones del laboratorio pudieron ser otro factor a tener en cuenta debido a que se desconoce si se siguieron todas las normas del laboratorio, la inexperiencia técnica del encargado de tomar las muestras es otro factor a tener en cuenta.

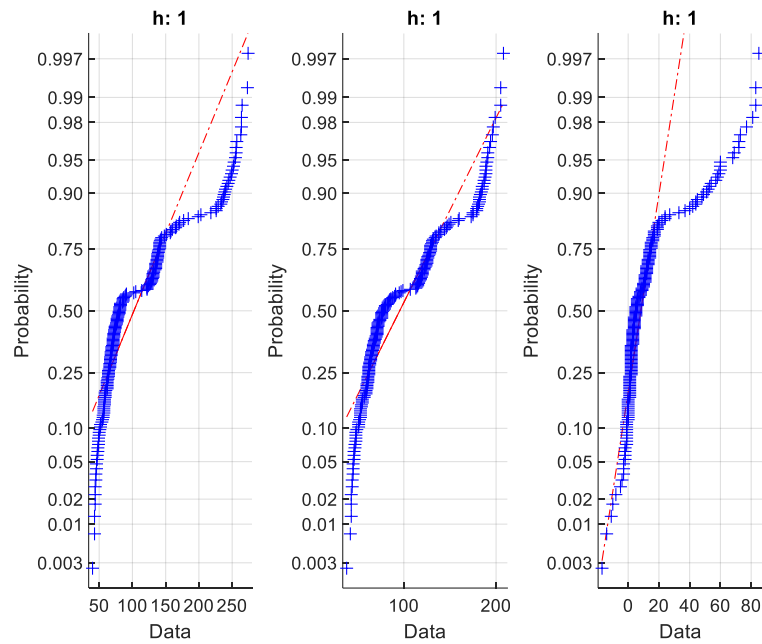


Figura 5.10 Comportamiento de hoja de normalidad, clase *Candida albicans* (M-Net V1 (Otsu))

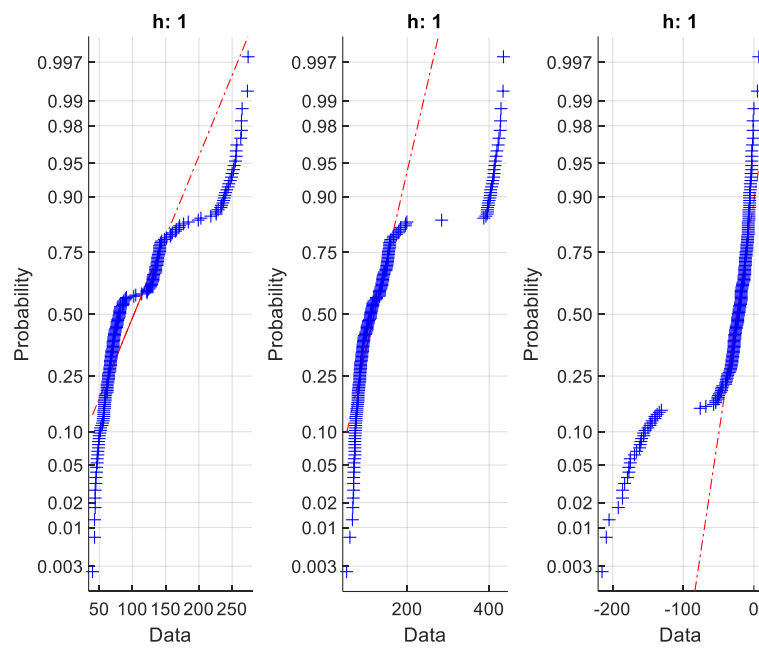


Figura 5.11 Comportamiento de hoja de normalidad, clase *Candida albicans* (M-Net V2 (Otsu))

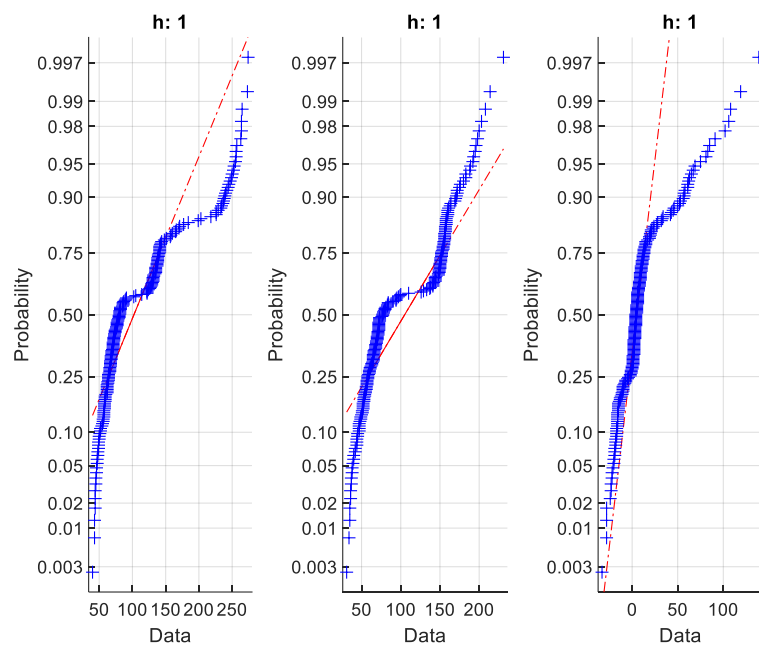


Figura 5.12 Comportamiento de hoja de normalidad, clase *Candida albicans* (M-Net V3 (Umbral))

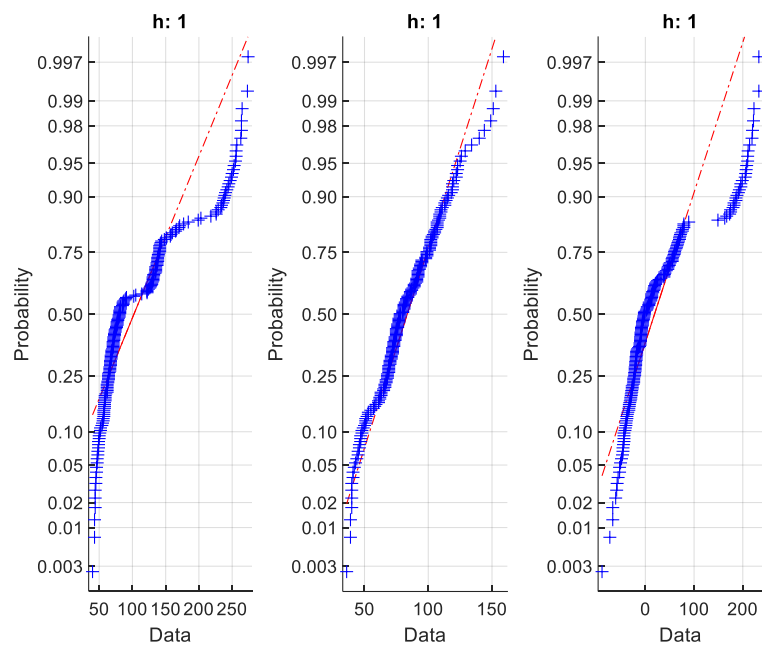


Figura 5.13 Comportamiento de hoja de normalidad, clase *Candida albicans* (MobileNet (Umbral))

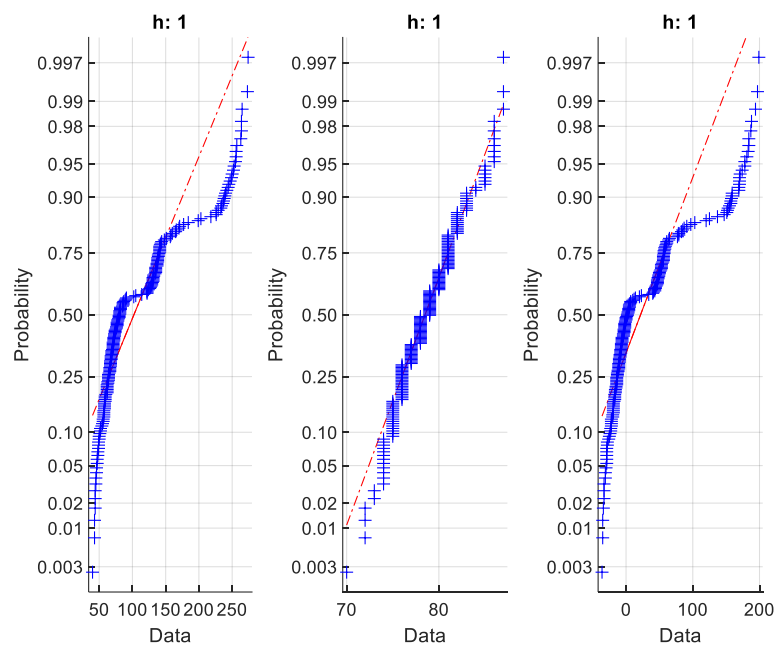


Figura 5.14 Comportamiento de hoja de normalidad, clase *Candida albicans* (ResNet (Umbral))

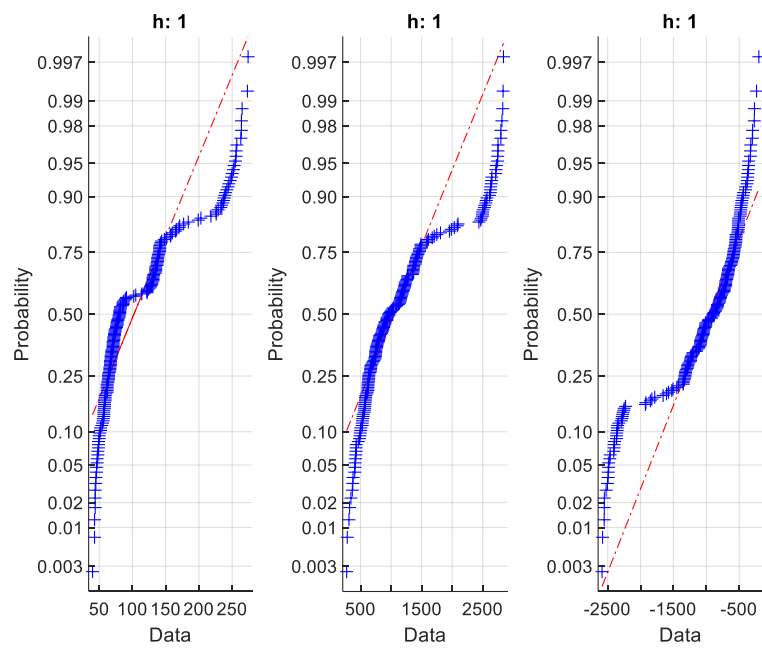


Figura 5.15 Comportamiento de hoja de normalidad, clase *Candida albicans* (U-Net (Umbral))

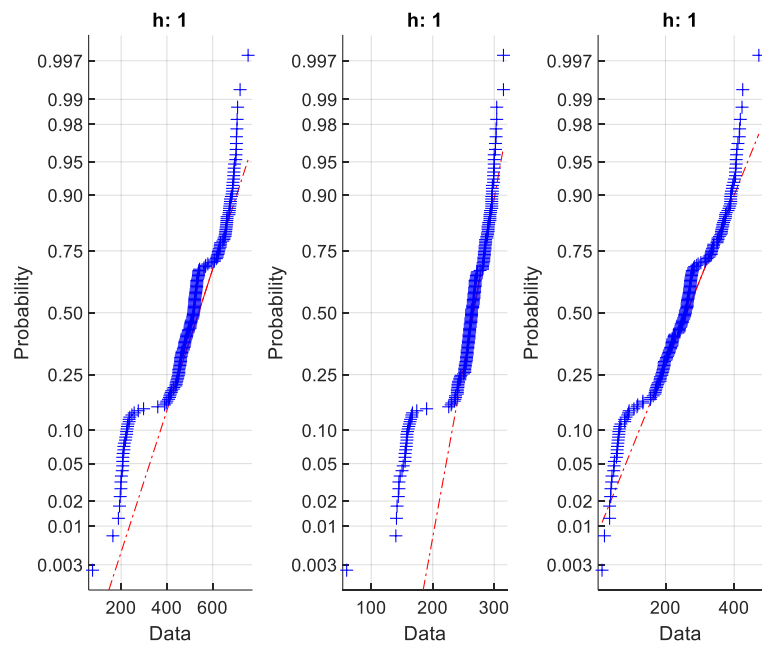


Figura 5.16 Comportamiento de hoja de normalidad, clase *Escherichia coli* (M-Net V1 (Otsu))

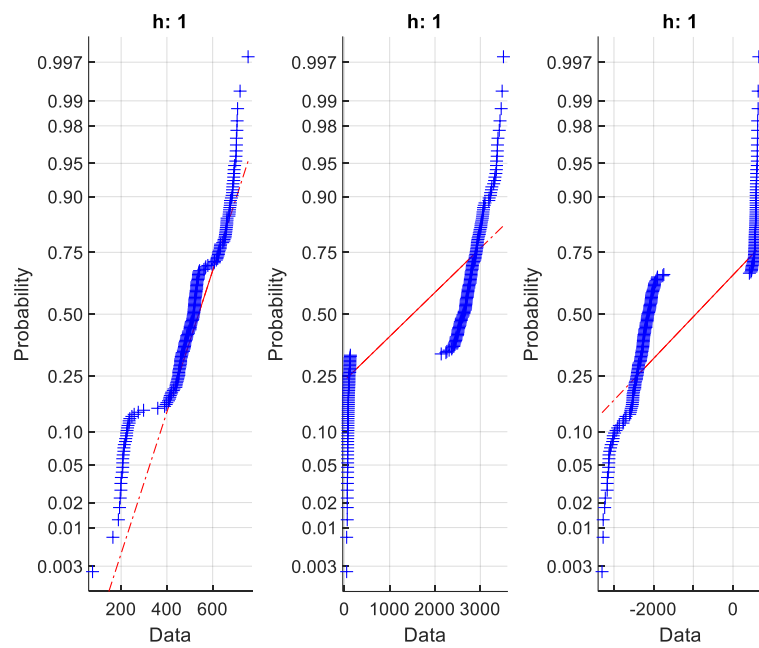


Figura 5.17 Comportamiento de hoja de normalidad, clase *Escherichia coli* (M-Net V2 (Otsu))

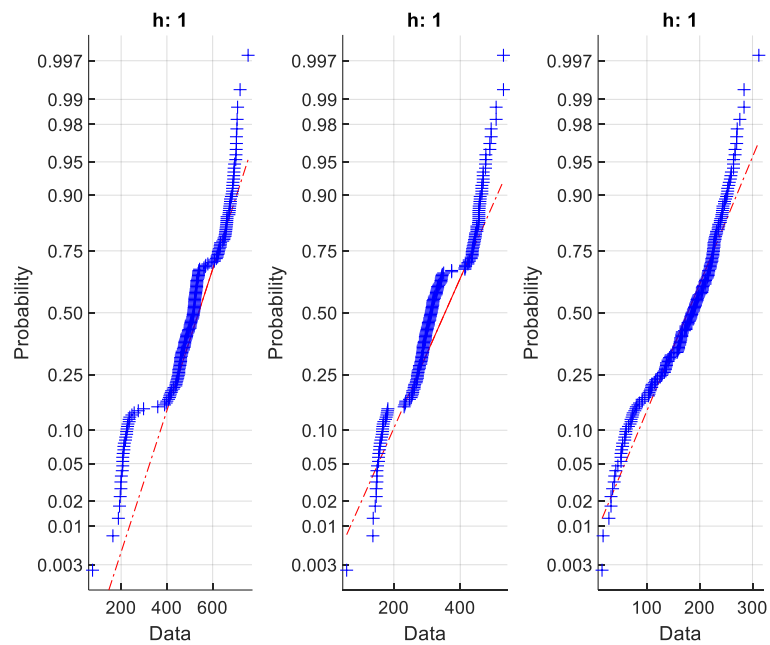


Figura 5.18 Comportamiento de hoja de normalidad, clase *Escherichia coli* (M-Net V3 (Umbra))

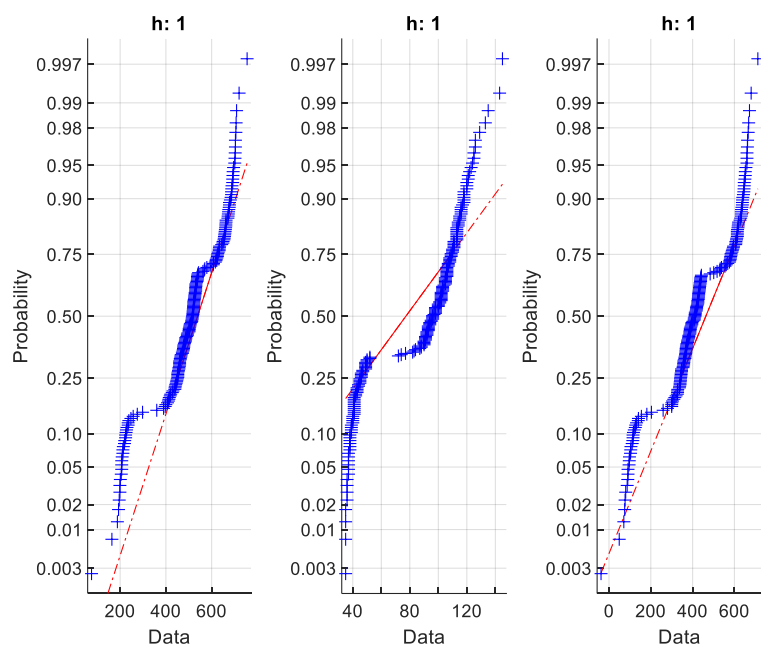


Figura 5.19 Comportamiento de hoja de normalidad, clase Escherichia coli (MobileNet (Umbral))

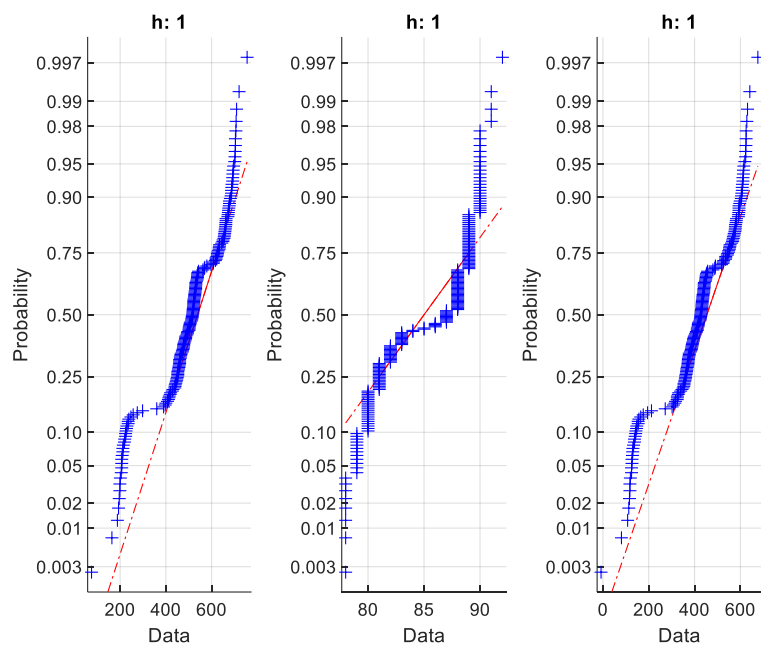


Figura 5.20 Comportamiento de hoja de normalidad, clase Escherichia coli (ResNet (Umbral))

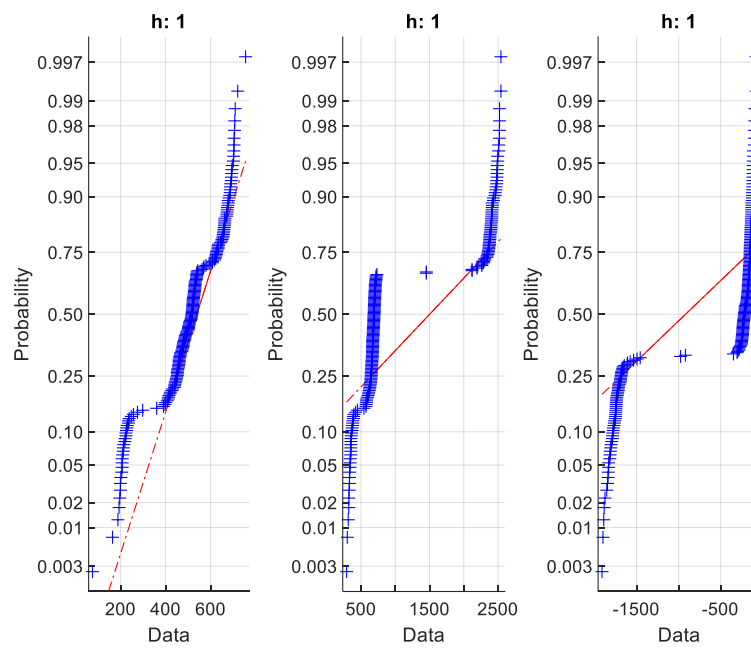


Figura 5.21 Comportamiento de hoja de normalidad, clase *Escherichia coli* (U-Net (Umbral))

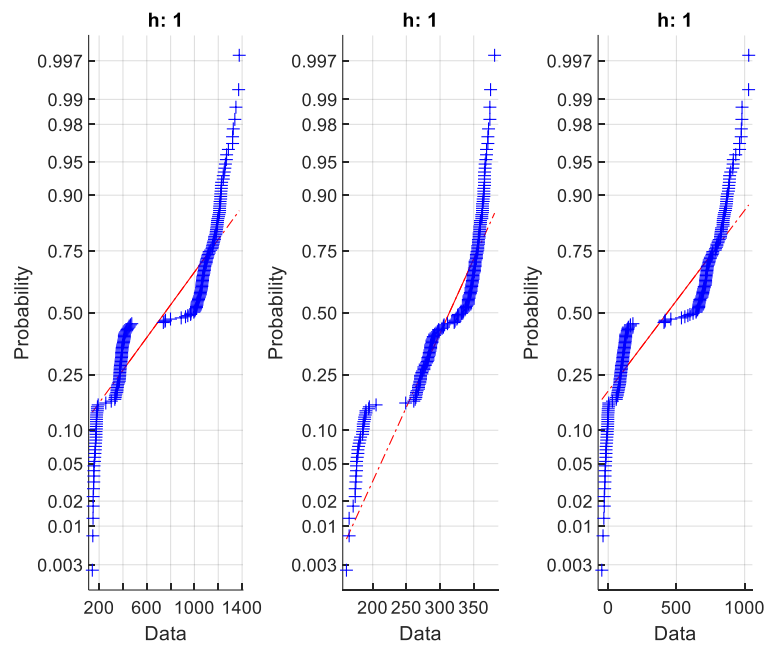


Figura 5.22 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (M-Net V1 (Otsu))

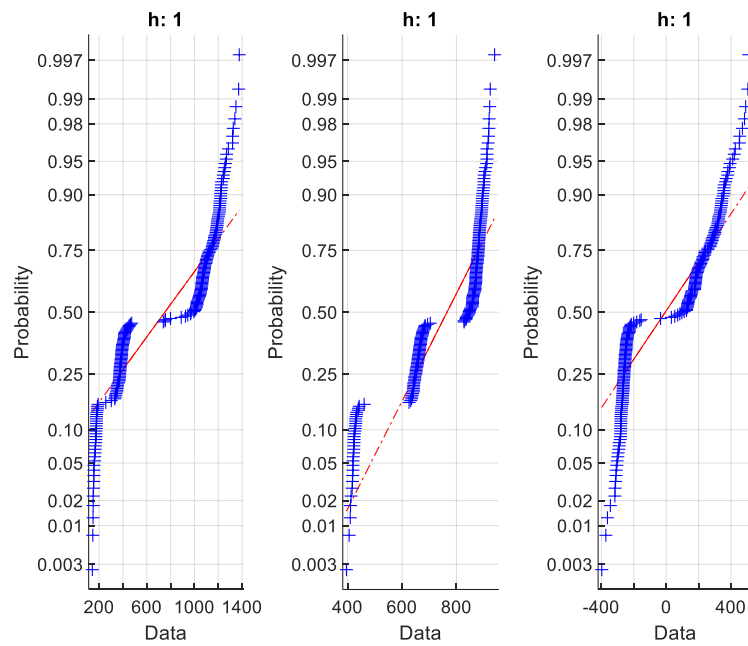


Figura 5.23 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (M-Net V2 (Otsu))

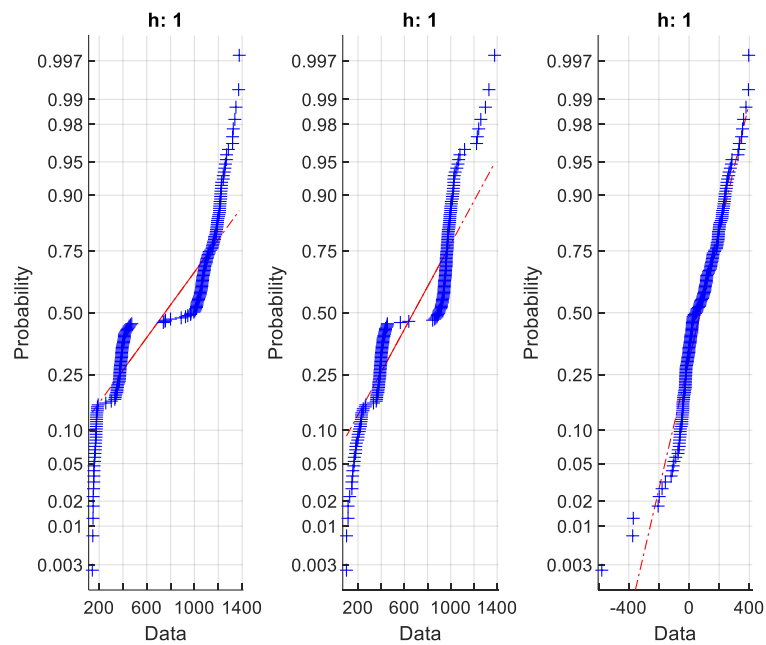


Figura 5.24 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (M-Net V3 (Umbral))

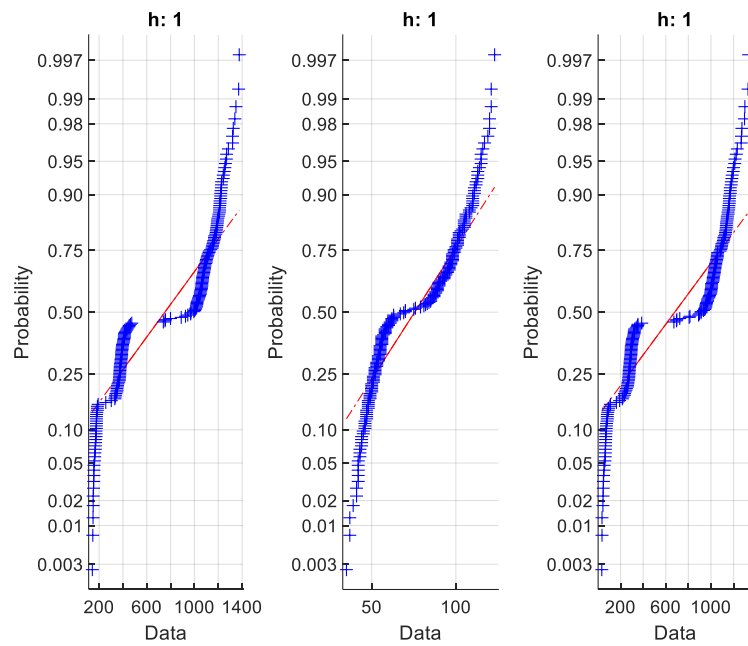


Figura 5.25 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (MobileNet (Umbra))

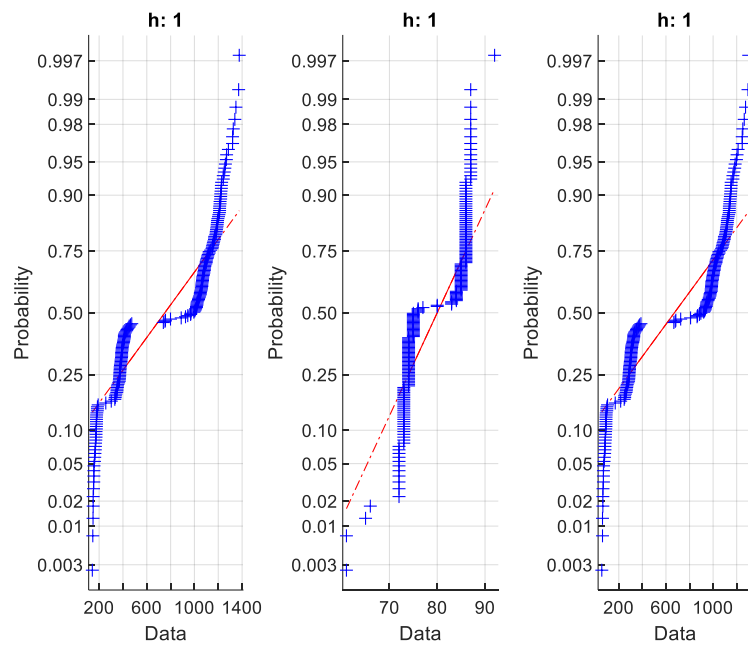


Figura 5.26 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (ResNet (Umbra))

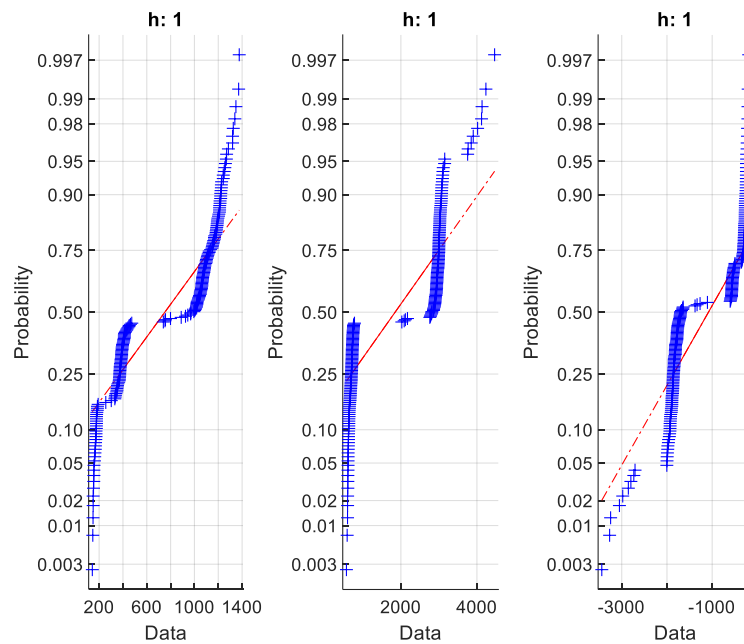


Figura 5.27 Comportamiento de hoja de normalidad, clase *Staphylococcus aureus* (U-Net (Umbral))

5.3.2 Prueba de Wilcoxon

La prueba de Wilcoxon con signo [49] tiene como hipótesis que dos conjuntos de datos son proporcionales o iguales. Al igual que Lilliefors, Wilcoxon tiene un parámetro α que indica si dicha hipótesis se confirma o se rechaza. De esta manera, si $\alpha \geq 0.05$, la hipótesis se confirma, de manera que las distribuciones son proporcionales. Por otro lado, si $\alpha < 0.05$, la hipótesis se rechaza, lo que quiere decir que existe significancia estadística entre ambas distribuciones. Para ejecutar la prueba de Wilcoxon, es necesario encontrar la diferencia de las variables aleatorias a estudiar dada por:

$$\Delta c = c_{FCN,l} - c_{GT,l} \quad (5.6)$$

Δc es la entrada al método de Wilcoxon y la salida de todas las pruebas realizadas indicaron que $\alpha=0$, lo que indica que existe significancia estadística entre los conteos obtenidos con $c_{FCN,l}$ y $c_{GT,l}$. Para comprender dicha significancia, se realizó un experimento que consiste en acomodar la cantidad de elementos contados por un humano de menor a mayor y graficarlo (Figura 5.28 grafica azul), luego comparar esa grafica con otra generada a partir de la cuenta de los elementos encontrados por un humano menos la cantidad de elementos encontrados por

el modelo sometido a la prueba (Figura 5.28 grafica roja), dando así una gráfica en la que es fácil visualizar la diferencia entre modelo así como la eficiencia de los mismos.

En este apartado se mostrarán las gráficas de las pruebas con el método Wilcoxon, las pruebas se generaron con los 6 modelos utilizados (las 3 versiones de M-Net, Mobilenet, Resnet y unet) con la intención de que el lector vea el comportamiento generado con cada modelo [51].

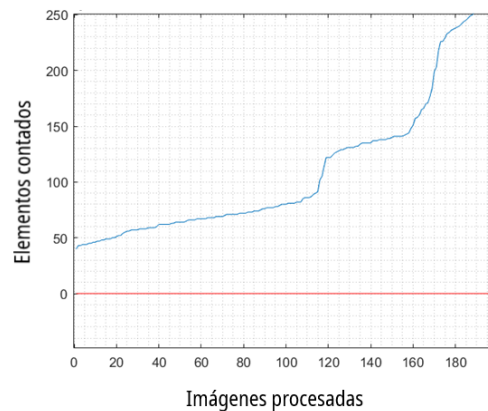


Figura 5.28 Resultado ideal que se busca obtener de los modelos utilizados con prueba Wilcoxon.

En la Figura 5.28 se muestra un ejemplo de cómo debería de ser una recta ideal, la gráfica de color azul representa el número de elementos identificados acomodados de menor a mayor, la gráfica de color rojo es la resta entre los elementos encontrados (gráfica azul) y los elementos que encontró el modelo, de forma ideal la gráfica de color rojo debería ser una constante en 0 (se espera una constante ya que la gráfica roja representa los elementos encontrados por un humano menos elementos encontrados por el modelo sometido a la prueba, y como se mencionó anteriormente de manera ideal el modelo debería encontrar la misma cantidad que el humano).

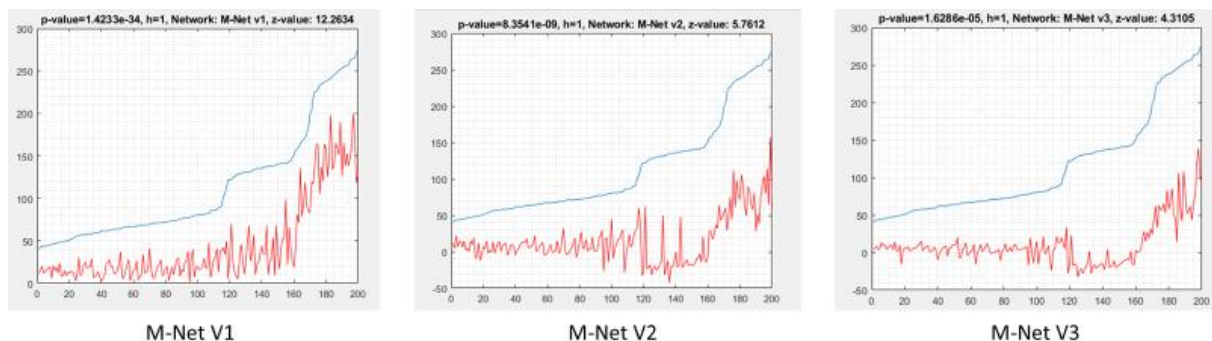


Figura 5.29 Comportamiento con clase *Candida albicans* (para las 3 versiones de M-Net)

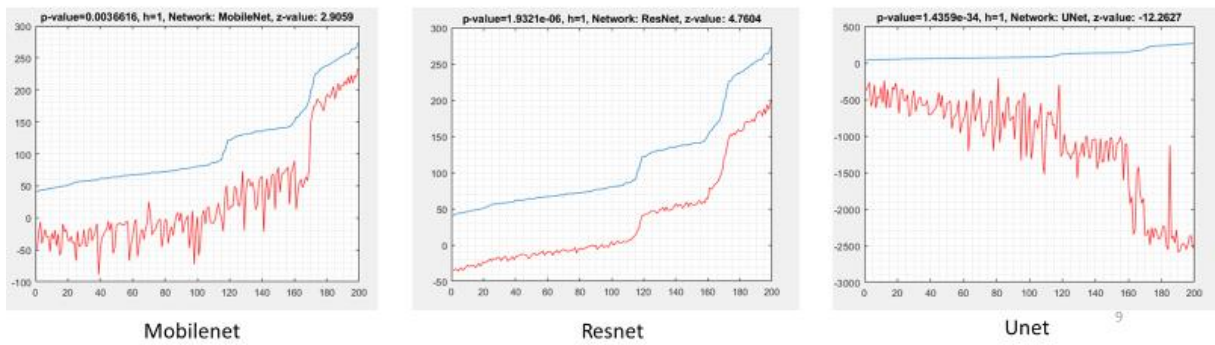


Figura 5.30 Comportamiento con clase *Candida albicans* (Modelos del estado del arte)

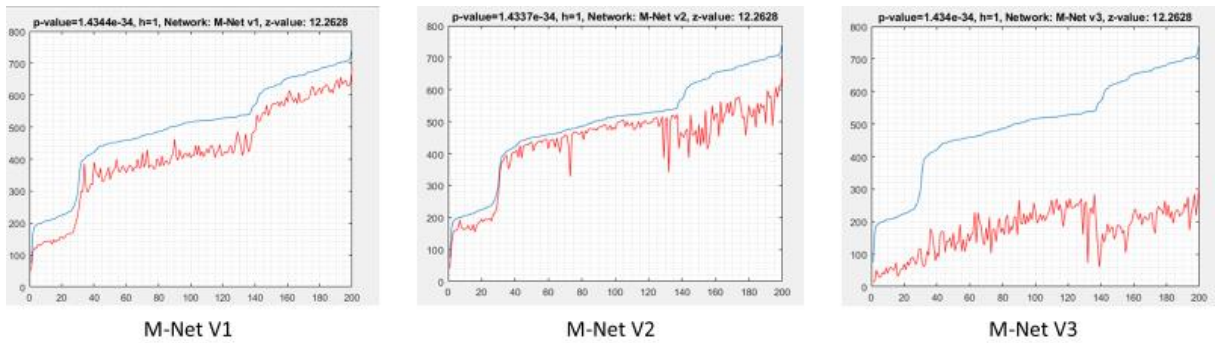


Figura 5.31 Comportamiento con clase *Escherichia coli* (para las 3 versiones de M-Net)

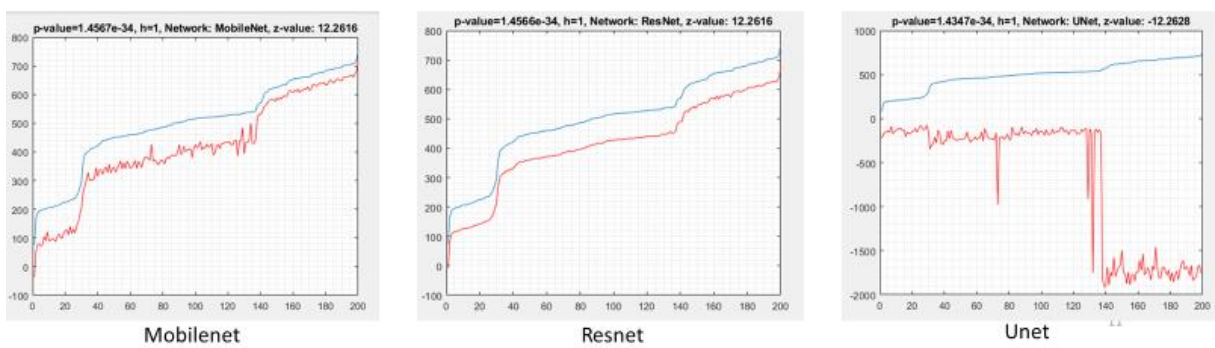


Figura 5.32 Comportamiento con clase *Escherichia coli* (Modelos del estado del arte)

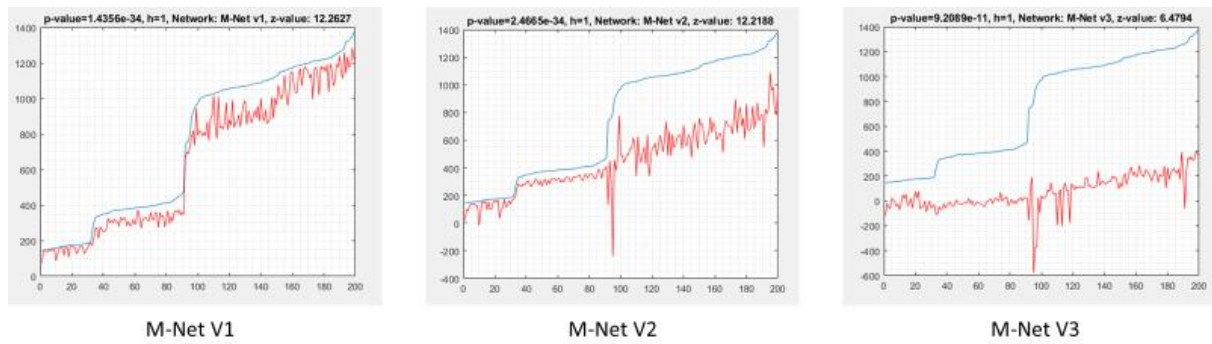


Figura 5.33 Comportamiento con clase *Staphylococcus aureus* (para las 3 versiones de M-Net)

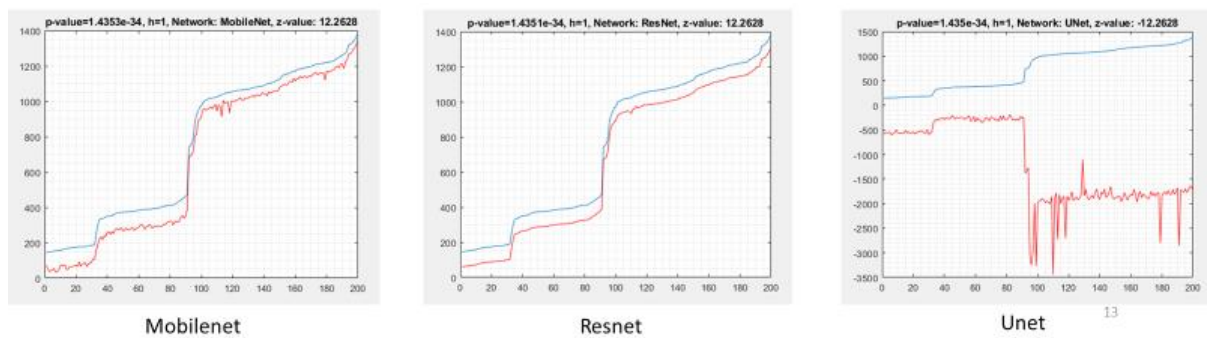


Figura 5.34 Comportamiento con clase *Staphylococcus aureus* (Modelos del estado del arte)

Se observa las 3 clases que nos muestran la Figura 5.29, Figura 5.30, Figura 5.31, Figura 5.32, Figura 5.33 y Figura 5.34 veremos que la que tiene un mayor acercamiento a la Figura 5.28 es el modelo M-Net V3, y las 3 clases tienen un comportamiento similar con respecto al crecimiento de elementos de interés de la imagen, conforme se elevan los elementos de interés la dificultad aumenta, el modelo comienza a tener complicaciones a la hora de agrupar los *blob* encontrados, debido a que una mayor cantidad de *blobs* representa eventualmente una cercanía entre ellos por lo que un conjunto de elementos de interés elevado que no está cerca a otro puede registrarse como tal durante la umbralización de la imagen generada por el proceso de predicción, a más elementos en una imagen menor cantidad de espacio para ser representados por lo que es natural que el mismo modelo agrupe conjuntos separados como si fuera un mismo elemento.

6 CONCLUSIONES

En esta tesis se presentó el diseño de una familia de redes totalmente convolucionales para detección y conteo de microorganismos denominadas M-Net. M-Net V1 tiene 14 capas convolucionales y 66025 parámetros. M-Net V2 tiene 10 capas convolucionales y 45484 parámetros. M-Net V3 tiene 12 capas (2 convolucionales, 6 convolucionales profundas y 4 convolucionales profundas separables) y 575 parámetros.

Los experimentos para diseñar y probar las tres versiones de M-Net se realizaron con la base de datos de Imágenes Reales de Bacterias y Levaduras (BYRI por sus siglas en inglés). Para el diseño de la base de datos, personal de la UACH generó 1502 imágenes de cultivos de las bacterias EC, SA y la levadura CA. De esta manera, se generaron 1502 imágenes con sus respectivos GTs. Entre estas imágenes, 500 son de EC, 506 SA y 496 CA.

6.1 Trabajo realizado

Los modelos de M-Net demostraron tener una arquitectura con buen desempeño y baja cantidad de parámetros. Esto se debe a que el diseño del autoencoder no comprime sucesivamente la imagen, el reducir las dimensiones espaciales de la imagen se usa para reducir el costo computacional y ayudar ligeramente al modelo a guardar la información relevante, aunado a esto el resultado que se necesita es una imagen binaria, si bien entrega resultados a escala de grises solo es necesaria utilizar un método de umbralización para obtener los resultados necesarios y procesarlos.

De acuerdo con las métricas de *F-score*, *Precision*, *Recall*, *IoU*, *Loss*, Parámetros, RAM y cuadros por segundo, el modelo M-Net V3 genera resultados igual de competitivos que sus versiones anteriores, pero reduce los parámetros más de un 98%. De acuerdo con la validación cruzada, las tres redes M-Net generalizan de manera consistente en diferentes subconjuntos de datos de entrenamiento y validación. Esto es un indicador positivo de que el modelo no está sobreajustando (*overfitting*) los datos de entrenamiento y que es capaz de manejar patrones relevantes tales como: fuentes de luz, fondos irregulares, elementos borrosos, elementos de interés pequeños, eliminación de artefactos entre otros. De esta manera, las redes M-Net pueden procesar correctamente imágenes nuevas.

En referente a los resultados cualitativos, se llevaron a cabo pruebas para comparar el desempeño de varios modelos de inteligencia artificial en una tarea particular relacionada con el conteo de microorganismos. La métrica de evaluación fue el conteo de microorganismos, que es una medida esencial en aplicaciones (como en las técnicas de citometría de flujo y de PCR) de biología y la microbiología. Para determinar cuál de los modelos era más eficaz en esta tarea, se aplicó el test de Wilcoxon, una técnica estadística que permite comparar pares de datos y determinar si hay diferencias significativas entre ellos. Con base en esta prueba, M-Net V3 logró los resultados más altos entre los modelos evaluados. Su capacidad para superar a sus propias versiones anteriores y a otros modelos sugiere que M-Net V3 es capaz de identificar y contar microorganismos de manera más precisa y efectiva.

6.2 Trabajo a futuro

Como trabajo futuro, se pueden realizar mejoras tales como obtener el centroide de los blobs detectados en las máscaras presentes en la base de datos utilizada para este proyecto asignar un único píxel para representar los microorganismos de interés. Esta idea busca optimizar el proceso de conteo de microorganismos, aumentando la rapidez, eficiencia y precisión.

El modelo utilizado en este proyecto de tesis es un *autoencoder* con redes neuronales totalmente convolucionales. Sin embargo, como parte de futuros desarrollos, se podría plantear la posibilidad de considerar redes tipo cápsula [52] [53] [54]. Esta elección se basa en la capacidad demostrada por el modelo actual para capturar características complejas en imágenes y segmentar objetos de interés, las redes tipo cápsula podrían generar resultados sumamente precisos y probablemente no unirían los *blobs* como uno solo, en pocas palabras puede tener un rendimiento considerable y reducir la cantidad de falsos positivos.

Por último, el siguiente paso en esta investigación es generar una interfaz gráfica para trabajar el modelo de una manera más accesible al público en general y se pueda cargar fácilmente en una computadora o como el modelo es sumamente ligero cargarlo en una página web para apelar más al concepto de telemedicina, realizar un reentrenamiento con el modelo con otras bases de datos para ver su adaptabilidad y como generaliza al plantearle un problema diferente, a fin de cuentas al ser un nuevo modelo es interesante ver que tanto se pueden explotar sus capacidades.

7 REFERENCIAS

- [1] J. Sereno, F. Bolaños, M. Vallejo, "Reconocimiento de Patrones Psicomotrices empelando Acelerometria y RNA", *ResearchGate*, p. 5, 2017.
- [2] H. A. Ríos et al, "RED NEURONAL EN LA DETECCIÓN DEL EDEMA MACULAR DIABÉTICO EN IMÁGENES DE FONDO DE OJO", *Revista Médica Sanitas*, vol. 20, nº 1, p. 10, 2017.
- [3] O. Juan Monroy-de-Jesús et al., "Clasificador de plantas medicinales por medio de deep learning", *Research in Computing Science* , pp. 65-78, 2019.
- [4] A. M. Lerma et al., "Clasificación automática de anastomosis mediante redes neuronales convolucionales en video fetoscópico", *Iberoamerica para la investigación y el desarrollo educativo*, vol. 12, nº 22, p. 25, 2021.
- [5] D. Morales-Venegas, S. Tello-Mijares, "Identificación de cáncer de mama en imágenes termográficas mediante redes neuronales convolucionales", *Revista Ciencia, Ingeniería y Desarrollo Tec Lerdo*, vol. 1, nº 6, p. 6, 2020.
- [6] I. Guyon, G. Dror, V. Lemaire, G. Taylor, D. Silver, "Autoencoders, Unsupervised Learning, and Deep Architectures", *Pierre Baldi*, vol. 27, p. 14, 2012.
- [7] W. A. Fonzi, et al., "Richard A. Calderone and William A. Fonzi", *TRENDS in Microbiology*, vol. 9, nº 7, p. 9, 2001.
- [8] F. Riverón et al., "RESISTENCIA BACTERIANA", 2003.
- [9] M. Castro, F. S. Rodriguez, "Levaduras: probióticos y prebióticos que mejoran la producción animal", *REVISTA CORPOICA* , vol. 6, nº 1, p. 13, 2005.
- [10] T.A.T. Gomes, W.P. Elias, I.C.A. Scaletsky, B.E.C. Gutha, J. Falcão, R.M.F. Piazzab, "Diarrheagenic *Escherichia coli*", *Brazilian journal of microbiology*, p. 28, 2016.
- [11] P. Garzón et al., "Staphylococcus aureus: generalidades, mecanismos de patogenidad y colonización celular", p. 14, 2019.
- [12] J. A. Ramírez, J. A. Parra, A. Alvarez, "Análisis de técnicas de recuento de Microorganismos.", *Programa de Microbiología. Universidad Libre Pereira*, p. 8.
- [13] S. Russell, P. Norving, "Inteligencia Artificial: Un Enfoque Moderno", PEARSON EDUCACIÓN, 2004.
- [14] C. Saavedra et al., "Redes Neuronales Artificiales".
- [15] R. S. Sutton et al., "Reinforcement Learning", London, England: The MIT Press, 2018.
- [16] S.-C. Wang, "Interdisciplinary Computing in Java Programming", Springer, 2003.
- [17] S.-Y. Huang, "MDPI", MDPI, 11 11 2022. [En línea]. Available: <https://www.mdpi.com/2075-4418/12/11/2765>. [Último acceso: 16 10 2023].
- [18] M García-Villanueva, L. Romero-Muñoz, "Diseño de una arquitectura de Red Neuronal Convolucional para la clasificación de objetos", de *Ciencia Nicolaita* #81, 2020.
- [19] M. C. B. REYES, "CLASIFICACIÓN DE MAMOGRAFÍAS MEDIANTE REDES NEURONALES CONVOLUCIONALES", *NUEVO LEÓN*, 2019.
- [20] A. Pandey, "Depth-wise Convolution and Depth-wise Separable Convolution", 9 Septiembre 2018. [En línea]. Available: <https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec>. [Último acceso: 23 Junio 2023].
- [21] M. Nielsen, "Neural Networks and Deep Learning", Yorktown Heights, NY, USA: Springer International Publishing AG, 2018.
- [22] D. Grattarola, D. Zambon, F. M. Bianchi, C. Alippi, "Understanding Pooling in Graph Neural Networks", *arXiv*, p. 21, 2021.

-
- [23] M. Blot, M. Cord, N. Thome, "MAX-MIN CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION", de Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005, Paris, 2016.
- [24] U. Muaz, "Autoencoders vs PCA: when to use ?", towards data science, 24 07 2019. [En línea]. Available: <https://towardsdatascience.com/autoencoders-vs-pca-when-to-use-which-73de063f5d7>. [Último acceso: 05 09 2023].
- [25] X.-X. Yin, L. Sun, Y. Fu, R. Lu, Y. Zhang, "U-Net-Based Medical Image Segmentation", *Journal of Healthcare Engineering*, p. 16, 2022.
- [26] K. Dong, C. Zhou, Y. Ruan, Y. Li, "MobileNetV2 Model for image classification", *IEEE Xplore*, p. 5, 2020.
- [27] A. Khan, A. Sohail, U. Zahoor, "A survey of the recent architectures of deep convolutional neural networks", *Springer Nature B.V.*, p. 62, 2020.
- [28] C. Shorten, "A survey on Image Data Augmentation for Deep Learning", *Journal of big data*, p. 48, 2019.
- [29] M. Bautista, P. A. Moreno, "Sistema de control de aparcamiento al aire libre mediante secuencias de imágenes", Jaén: Escuela Politécnica Superior de Linares, 2018.
- [30] N. K. R. G. Dor Bank, "Autoencoders", *arXiv*, p. 22, 3 Abril 2021.
- [31] B. Al, "Intro a las redes neuronales convolucionales", 23 Noviembre 2019. [En línea]. Available: <https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8>. [Último acceso: 01 Agosto 2023].
- [32] Wikipedia, "Método del valor umbral", Wikipedia, 16 Enero 2023. [En línea]. Available: https://es.wikipedia.org/wiki/M%C3%A9todo_del_valor_umbral. [Último acceso: 01 Agosto 2023].
- [33] D. S. Chen, "A Robust Back Propagation Learning Algorithm for Function Approximation", *IEEE*, vol. 5, nº 3, p. 13, 1994.
- [34] R. C. Gonzalez, "Digital Image Processing", New York: Pearson, 2018.
- [35] T. D. Science, "Focal Loss: A better alternative for Cross-Entropy", 26 Abril 2022. [En línea]. Available: <https://towardsdatascience.com/focal-loss-a-better-alternative-for-cross-entropy-1d073d92d075>. [Último acceso: 22 junio 2023].
- [36] R. Nayak, "towardsdatascience", 22 04 2022. [En línea]. Available: <https://towardsdatascience.com/focal-loss-a-better-alternative-for-cross-entropy-1d073d92d075>. [Último acceso: 22 08 2023].
- [37] Wikipedia, "Graphics processing unit", Wikipedia, 7 Agosto 2023. [En línea]. Available: https://en.wikipedia.org/wiki/Graphics_processing_unit. [Último acceso: 10 Agosto 2023].
- [38] Stanford, "risc vs cisc", Stanford, [En línea]. Available: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/>. [Último acceso: 10 Agosto 2023].
- [39] Guru99, "Difference Between RISC and CISC processor", Guru99, 2023. [En línea]. Available: <https://www.guru99.com/risc-vs-cisc-differences.html>. [Último acceso: 10 Agosto 2023].
- [40] N. Corporation, "nvidia DEVELOPER", Nvidia, 2023. [En línea]. Available: <https://developer.nvidia.com/embedded/jetson-tx2>. [Último acceso: 22 08 2023].
- [41] C.-T. & UniTN, "CPU performance", CNR-TN & UniTN, 11 Agosto 2023. [En línea]. Available: https://gene.disi.unitn.it/test/cpu_list.php. [Último acceso: 11 Agosto 2023].
- [42] O. S. Quesada, "Optimización de redes neuronales para ejecución eficiente de aplicaciones de inteligencia artificial en GPUs embebidos.", Costa Rica: Instituto Tecnológico de Costa Rica, 2021.
- [43] A. Shridhar, "Interoperating Deep Learning models with ONNX.jl", *Juliacon*, p. 6, 2019.
- [44] NVIDIA, "Performance optimization help", NVIDIA, 18 11 2018. [En línea]. Available: <https://forums.developer.nvidia.com/t/performance-optimization-help/67567>. [Último acceso: 24 8 2023].
- [45] M. Qasimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, P. H. Jones, "Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels", *IEEE*, p. 9, 2019.
-

- [46] F. Mazzocchi, P. Bendedicte, H. Tabani, L. Kosmidis, J. Abella, F. J. Carzara, "*Performance Analysis and Optimization of Automotive GPUs*", *IEEE*, p. 8, 2019.
- [47] J. D. S, "*INFERENCIA ESTADÍSTICA: PRUEBAS DE HIPÓTESIS*", 2014.
- [48] statologos, "*Prueba de Lilliefors para normalidad y distribuciones exponenciales*", statologos, 2023. [En línea]. Available: <https://statologos.com/prueba-de-lillie-fors/>. [Último acceso: 03 09 2023].
- [49] D. Team, "*Prueba de Wilcoxon*", DATAtab, 2023. [En línea]. Available: <https://datatab.es/tutorial/wilcoxon-test>. [Último acceso: 03 09 2023].
- [50] Romero-Saldaña, "*Pruebas de bondad de ajuste a una distribución normal*", *Revista Enfermería del Trabajo*, p. 10, 2016.
- [51] V. B. Silvente, "*Clasificación de pruebas no paramétricas*", *REIRE*, vol. 5, nº 2, p. 13, 2012.
- [52] S. Yang, F. Lee, R. Miao, J. Cai, L. Chen, W. Yao, K. Kotani, Q. Chen, "*RS-CapsNet: An Advanced Capsule Network*", p. 12 2020.
- [53] S. Toramana, "*Convolutional capsnet: A novel artificial neural network approach to detect COVID-19 disease from X-ray images using capsule networks*", *elsevier*, p. 11, 2020.
- [54] H. Xiang, "*3-D Res-CapsNet convolutional neural network on automated breast ultrasound tumor diagnosis*", 2021.